Self-Organization in Networks of Mobile Sensor Nodes

Dissertation

zur Erlangung des akademischen Grades
doctor rerum naturalium (Dr. rer. nat.)
im Promotionsfach Informatik
eingereicht an der

Mathematisch-Naturwissenschaftlichen Fakultät der
Humboldt-Universität zu Berlin

von Dipl.-Phys. Christian Blum

Präsident der Humboldt-Universität zu Berlin:
Prof. Dr. Jan-Hendrik Olbertz

Dekan der Mathematisch-Naturwissenschaftlichen Fakulät:
Prof. Dr. Elmar Kulke

Gutachter(innen):

1. Prof. Dr. Verena V. Hafner (Humboldt-Universität zu Berlin)
2. Prof. Dr. Hans-Dieter Burkhard (Humboldt-Universität zu Berlin)
3. Prof. Alan F. T. Winfield, PhD (University of the West of England)

Eingereicht am: 12. August 2015
Tag der Verteidigung: 27. November 2015

Zusammenfassung

Selbstorganisierte drahtlose (ad hoc) Multihopnetzwerke können als einfach einsetzbare, robuste und rekonfigurierbare Kommunikationsinfrastruktur eingesetzt werden, die zum Beispiel in Katastrophenszenarien genutzt werden können wenn die statische Kommunikationsinfrastruktur ausgefallen ist. Die Verbreitung dieser Art von Netzwerken wird zudem durch die fortschreitende Vernetzung von unterschiedlichsten Objekten im Zuge des Internet der Dinge weiter vorangetrieben. Hierbei werden (Gebrauchs-) Gegenstände durch meist drahtlose Netzwerkschnittstellen erweitert was sie in die Lage versetzt drahtlose (Sensor-) Netzwerke aufzuspannen. Solche drahtlosen Netzwerke nutzen die Luft als geteiltes Kommunikationsmedium was bedeutet, dass die Parameter des Netzwerks sehr orts- und zeitveränderlich sowie verrauscht sind. Intelligente Roboter können in der Rolle von Netzwerkknoten diese Herausforderungen bewältigen indem sie sensomotorische Interaktion ausnutzen. Dies heißt, dass sie die Sensorinformation aktiv durch ihre Bewegung in der drahtlosen Umgebung formen um dann die Zusammenhänge zwischen Bewegung und Sensorwerten auszunutzen und so die Komplexität zu reduzieren.

Die Problemstellung für diese Arbeit ist daher eine Kontrollstrategie für autonome Roboter zu entwerfen, die die Roboter in die Lage versetzt sich in drahtlosen (ad hoc) Multihopnetzwerken zu integrieren, die nur auf lokaler Information basiert. Diese lokalen Kontrollstrategien für jeden einzelnen Roboter koppeln im Falle mehrere Roboter so die einzelnen Roboter lose durch die jeweiligen sensomotorischen Schleifen. Diese lose Kopplung führt dann zu globaler Selbstorganisation des Systems.

Beispielhafte Messungen von Netzwerkparametern wie zum Beispiel Signalstärke wurden durchgeführt um ein Grundverständnis echter Netzwerkdynamiken als Basis für weitere Experimente und für die Entwicklung von Algorithmen zu vermitteln. Für diese Messungen wurden verschiedene Roboterplattformen, ein händisch getragener Laptop sowie ein Spektrumanalysator benutzt um jeweils unterschiedliche Aspekte der Netzwerkdynamiken zu untersuchen.

Für einzelne Roboter wurde ein Algorithmus zur Exploration unbekannter Netzwerke für großflächigen Außeneinsätze entwickelt und in Simulation evaluiert, wobei sich der Algorithmus als sehr tolerant gegenüber Messrauschen erwiesen hat. Des Weiteren wurde ein gradientenbasierter Navigationsalgorithmus vor allem für das Lokalisieren anderer Netzwerkknoten entwickelt. Die Konvergenzkriterien dieses Algorithmus wurden analytisch bestimmt und der Algorithmus beispielhaft auf einem Roboter implementiert und experimentell in einem Innenraumszenario evaluiert. Es wurde auch gezeigt wie diese Art von Algorithmen sich auf andere Aufgaben als das Finden anderer Netzwerkknoten erweitern lassen. Zusätzlich wurde ein Metaalgorithmus basierend auf internen Modellen, der Aufgaben, wie zum Beispiel das Finden eines Netzwerkknotens oder die überbrückung zweier Netzwerkknoten lösen kann, entwickelt, implementiert und experimentell in einem Innenraumszenario evaluiert.

Für den Fall von mehreren Robotern, d.h., ein Schwarm drahtlos verbundener Roboter, wurden bereits viele interessante Algorithmen, wie zum Beispiel für die optimale Platzierung mobiler Netzwerkknoten, in der Literatur vorgestellt. Allerdings wird im Moment keiner dieser Algorithmen in realen Szenarien eingesetzt. Einer der Gründe hierfür ist, dass naive Sachwarmalgorithmen meist nicht sicher genug für reale Anwendungen und außerdem meist nicht tolerant gegenüber (Teil-) Ausfällen von Robotern sind. Um dieses Problem zu lösen wurde der Einsatz einer Architektur, basierend auf internen Modellen die einen internen Simulator nutzen, untersucht um die Sicherheit und Fehlertoleranz dieser Systeme zu erhöhen und sie damit geeigneter für reale Anwendung zu machen.

Für diese Architektur wurden zwei Testszenarien untersucht: Im ersten Experiment musste ein Roboter andere Roboter davor bewahren zu Schaden zu kommen und dabei gleichzeitig seine eigene Sicherheit gewährleisten. Zusätzlich hatte er dabei eine Aufgabe zu erfüllen. In einem zweiten Experiment musste ein Roboter durch einen engen Korridor navigieren ohne mit anderen Robotern zusammenzustoßen. Die vorgeschlagene Architektur hat sich in beiden Experimenten als sehr effektiv erwiesen.

Abstract

Self-organized wireless multihop (ad hoc) networks can form an easily deployable, robust and reconfigurable communication infrastructure, which can be employed for example in a disaster scenario, when static communication infrastructure has been destroyed. Furthermore, the paradigm of the Internet of things will add network capabilities to many objects, often via wireless interfaces enabling machine-to-machine communication creating a wireless (sensor) network. As such wireless networks use the air as a shared physical medium, the parameters of these networks often show very space- and time-varying noisy characteristics. Intelligent robotic network nodes can overcome these problems posed by these dynamics and measurement noise by exploiting sensorimotor interaction. By actively shaping the sensory information by moving the robot in the wireless environment, complexity can be reduced.

Exemplary measurements of network parameters such as signal strength have been performed to form an experimental foundation for the design of algorithms for robots integrating into wireless networks. For these measurements, robot platforms, a manually carried laptop, as well as a stationary spectrum analyzer have been employed to target different aspects of the network dynamics.

For single robots, an algorithm for network exploration of unknown networks for large scale outdoor scenarios has been developed and evaluated in simulation. This algorithm has proved to be very tolerant against measurement noise. Furthermore, a gradient-based algorithm for navigation in wireless networks, specifically for the task of locating another network node has been developed. The convergence conditions of this algorithm have been derived analytically for the case of a robot navigating a wireless network. This algorithm has also been implemented on a real robot and evaluated experimentally in an indoor scenario. Additionally, it was shown how this algorithm can be extended to other tasks than locating a network node. In addition to these algorithms, an internal model-based meta-algorithm to solve tasks like locating a node or bridging two nodes has been developed and implemented on a different real robot and evaluated experimentally in an indoor scenario.

For the case of multiple robots, i.e., a swarm of wirelessly connected robots, a lot of interesting algorithms for example for optimal placement of mobile network nodes have been developed. However, none of them are being used in real world scenarios. One of the reasons for that is insufficient safety and fault tolerance of naive algorithms. To overcome this problem the use of an internal model-based architecture using an internal simulation has been investigated in multi-robot scenarios to enhance the safety of these systems, making them more suitable for real world use. Two test scenarios have been investigated using this architecture: in the first experiment a robot had to prevent other robots from coming to harm while ensuring its own safety and fulfilling a task. In a second experiment, a robot had to navigate through a narrow corridor without colliding with other robots. The proposed architecture has been shown to be very effective.

This work was created using the typesetting system LaTeX in combination with the standard scrbook class from the KOMA-Script bundle. Figures and schematics have been created using Inkscape, Gimp, Adobe® Photoshop® Lightroom® software, Microsoft® Image Composite Editor software and matplotlib. All pictures from other sources have been declared as such.

Preface

I first came into contact with statistical methods, in particular machine learning methods for data analysis during my physics studies, which sparked my interest in the field of artificial intelligence. Specifically I was interested in neural networks and the question of how cognition in general, especially in biological systems, but also in artificial systems, could work.

As a consequence after finishing my physics studies I wanted to work on a topic related to artificial intelligence. While looking for a position as a PhD student, I quickly realized that most of the current research on artificial intelligence involves robots. I was lucky enough to get a position in Prof. Hafner's research group at Humboldt-Universität zu Berlin, funded by the graduate school Model-Based Development of Technologies for Self-Organizing Decentralized Information Systems in Disaster Management (METRIK).

Because I am originally a physicist by trade rather than a computer scientist, I was accepted for a qualification scholarship, which allowed me to fill some knowledge gaps in computer science and also got me first into contact with robotics research. During that time I also was able to participate in the excellent ShanghAI lectures[1], which were instructional in why cognition and artificial intelligence research is done with robots. It was at this time as I began to see robotics as more of a synthetic method to understand cognition in biological systems, which is today mainly driving my interest in robotics.

The other main inspiration at the beginning of my research was the graduate school METRIK. Since it mainly deals with self-organized decentralized information systems in disaster scenarios, my involvement lead to the work on network robotics. This area is, besides being an interesting testbed scenario for general robotics questions, therefore also of practical technical relevance in the context of the graduate school.

Swarm research was interesting to me well before starting my PhD, which the focus of my graduate school on self-organizing decentralized systems reinforced. In particular, I also helped to organize the swarm seminar held by Prof. Hafner for four semesters, which brought me into contact with a large part of the swarm literature and thus also with the current research in the field. At the beginning of my research I performed a number of toy simulations related to swarm algorithms, mainly with a focus on pattern formation. Although they were instructional in my understanding of self-organized algorithms, they never led to any publications. During that time I also briefly visited the Bristol Robotics Laboratory (BRL) and first met Prof. Winfield and came into contact with some of the work and infrastructure of his swarm robotics group. At this point I did not investigate the subject further as I did not have access to the necessary infrastructure to perform swarm experiments going beyond the toy simulations.

[1] http://shanghailectures.org/

In order to get some understanding of the dynamics of wireless networks in real world scenarios, I began working on measurements of network parameters, especially physical ones such as signal strength. Initially I believed that algorithms from vision could be adapted to network robotics , in particular from computer vision as well as biologically inspired vision algorithms. However, during my initial experiments and measurements I realized that gradient based algorithms work better and are more simple as well as lending themselves better to theoretical analysis. Furthermore, I worked on a simple and robust algorithm for network exploration, which was only implemented in simulation due to the lack of a working robotic platform for large-scale outdoor experiments.

At the same time I began working on flying robots as a robotic platform for these algorithms. I chose a flying platform because large scale effects in wireless networks can be on length scales of tens and hundreds of meters and flying robots can operate on those distances without having to deal with obstacle avoidance and with only minimal path planning. Furthermore, they can operate in line-of-sight conditions to the network nodes most of the time, which simplifies the complex network dynamics. However due to the time and resources required to gain proficiency with flying robots, in particular piloting, ground-based robots were chosen instead.

To this end I began working on gradient-based algorithms for ground-based robots with a focus in theoretical convergence analysis. Using a ground-based robot also meant performing indoor experiments because of the practicalities of dealing with the complex dynamics of indoor wireless networks. This also lead to the later experiments in the network robotics context based on internal models as I believed they cope better with these complex dynamics. Furthermore, the idea of encoding complex goals using network parameters was first explored in that work.

At that time I also learned about the concept of internal models, mainly through the work done in our group on humanoid robots in the context of sensorimotor exploration, sensorimotor learning and behavior recognition. During my work on network robotics, I realized that a lot of algorithms for self-organization exist in literature but they are very rarely used outside of simulations, i.e., with real robots.

After discussing possible topics on swarm robotics for a research visit with Prof. Winfield, it was agreed that I would work on an internal simulation based consequence engine. This allowed me to combine my interests in swarm robotics with internal models, and was also attractive for our group. The use of an off-the-shelf simulator, including several robots as well as the environment for the internal simulation complemented the focus of our group on learned paired inverse-forward body models. I applied for Deutscher Akademischer Austausch Dienst (DAAD) funding for a research visit, which was accepted and during which most of the work presented in the last part of this thesis on self-organization was conducted.

After returning to Berlin, I made a final attempt at working with flying robots, which again proved impossible due to time constraints. Instead I decided to implement an internal model based approach in the network robotics context for a ground-based robot in an indoor scenario. This work also furthered the earlier idea of encoding complex goals using network parameters and experimentally showed the feasibility of this approach. After finishing the design, implementation and experiments, I began working on this thesis.

I would like to thank everybody who has made my research and this work possible, in particular I would like to thank Prof. Dr. Verena V. Hafner for granting the opportunity to work in her group and acting as my first supervisor and reviewer, Prof. Dr. Alan F. T. Winfield for hosting my visit to the Bristol Robotics Lab and acting as my reviewer, Prof. Dr. Joachim Fischer as my second supervisor, Prof. Dr. Hans-Dieter Burkhard as my my reviewer, Antonio, Bruno, Carsten, Claas, Damien, Ferry, Guido, Heinrich, Marcus, Maria Elena, Michael, Oswald, Philipp, Sasa, Siham, the rest of the Adapt group, the Nao Team Humboldt, the members of GRK METRIK, Robert and Anatolij for fruitful discussions, Gabriele Graichen, Renate Zirkelbach, and Sabine Becker, all the people proof reading thesis and especially Janina and Mike, and all my colleagues, friends, and my family for your support and suggestions.

I deeply apologize to anybody who was missed here.

I would also like to thank the organizations who funded me during my research. In particular I want to thank the DFG graduate research training group METRIK (GRK 1324) for funding. I am also grateful to the Deutscher Akademischer Austausch Dienst (DAAD) for supporting me while visiting researcher at the BRL.

DAAD Deutscher Akademischer Austausch Dienst
German Academic Exchange Service

Berlin, July 2015 Christian Blum

Contents

Abbreviations

AE Action Evaluator

AG Action Generation

AIS Artificial Immune System

AS Action Selection

BER Bit-Error Rate

BFGS Broyden-Fletcher-Goldfarb-Shanno

BRL Bristol Robotics Laboratory

BRL Bristol Robotics Laboratory

CE Consequence Engine

DAAD Deutscher Akademischer Austausch Dienst

DOF degrees of freedom

FDI fault detection and isolation

FDSA Finite Difference Stochastic Approximation

FDTD finite-difference time-domain

FWHM full width at half maximum

GPS Global Positioning System

GSM Global System for Mobile Communications

HWL Humboldt Wireless LAB

IMU internal measurement unit

ISM Industrial, Scientific and Medical

ISO International Organization for Standardization

ITU International Telecommunication Union

k-NN k-Nearest Neighbors

LLC Logical Link Control

MAC Media Access Control

MCMC Markov chain Monte Carlo

MEEP MIT Electromagnetic Equation Propagation

METRIK Model-Based Development of Technologies for Self-Organizing Decentralized Information Systems in Disaster Management

MIMO multiple-input and multiple-output

MLP Multilayer Perceptron

MSE mean squared error

OSI Open Systems Interconnection

OTL Object Tracker-Localiser

PCA principal component analysis

PCB printed circuit board

PDR packet delivery rate

PER packet error rate

PF potential function

PHY physical layer

PID proportional-integral-derivative

RC Robot Controller

RDSA Random Direction Stochastic Approximation

RSSI Received Signal Strength Indication

SAR specific absorption rate

SEL Safety/ethical Logic

SLAM Simultaneous Localization and Mapping

SNIR signal-to-noise plus interference

SNR signal-to-noise ratio

SPP self-propelled particle

SPSA Simultaneous Perturbation Stochastic Approximation

SVM Support Vector Machine

TCP Transmission Control Protocol

UDP User Datagram Protocol

UMTS Universal Mobile Telecommunications System

WLAN Wireless Local Area Network

WSN Wireless Sensor Network

Chapter 1

Introduction

1.1. Motivation

Network robotics deals with mobile autonomous nodes, which form or are part of a (wireless) network. The practical technical relevance of network robotics is comprehensive: robots need to connect to and interact with wireless (ad hoc) networks for a variety of reasons, ranging from interacting with smart objects in the context of the Internet of things, over getting data from sensor networks, or cloud resources, to interaction with humans through wirelessly connected terminals, or wearable devices. For a more in-depth discussion on this topic, refer to section 4.1. Furthermore, robots can employ wireless communication to exchange information between each other to coordinate joint tasks and can also be used as mobile nodes in wireless networks to improve or repair the networks. These possibilities are further discussed in section 9.3. A network consisting purely of mobile nodes can also form a very flexible and quickly deployable network infrastructure, for example needed in disaster scenarios when the stationary infrastructure is out of order or destroyed.

Integrating into or forming a wireless network means at the most basic level — besides technical issues like protocols etc. — establishing a good reception of the wireless signals emitted by other network nodes. For a single mobile network node this means movement as it is the only option at its disposal to influence reception assuming an omni-directional antenna. Further tasks for a mobile network node include finding other network nodes or reducing interference to other network nodes. Network robotics therefore means for a single robot navigation in the wireless network formed by either stationary or other mobile nodes.

A robot navigating a wireless network can measure network parameters such as signal strength as a sensory input, usually as a scalar field. Navigation of robots sampling scalar fields as measurement inputs has long been a field of research in robotics, inspired by for example bacterial chemotaxis, during which bacteria sample chemical concentrations. However, in contrast to chemotaxis, experimentation environments are readily available in the form of ubiquitous wireless networks. This facilitates implementations on real robots because no special experimental environment is necessary. At the same time, algorithms based on scalar measurements are typically simple enough to be fully embodied in the robot itself, which is especially important for small and/or flying robots.

Wireless networks, more specifically measuring their parameters as sensory modalities, lead to much more challenging dynamics than normally encountered when sampling scalar

fields, varying on length scales of typical robots as well as quickly varying temporally (for a more in-depth discussion see chapter 2). The spatial variations are mainly due to the physical characteristics of electromagnetic waves because reflections lead to (self-) interference between the wave and its reflections leading to variations on the scale of the wavelength of the electromagnetic wave ($12.5\,\mathrm{cm}$ for $2.4\,\mathrm{GHz}$). Additionally, a lot of objects in the environment such as walls attenuate the electromagnetic wave. The temporal variations are mainly due to the fact that spectral bands commonly used in wireless networks are unlicensed. This means that other users using the same frequency bands and potentially different protocols and communication technologies can lead to external interference. This external interference shows temporal variance because of the activity patterns of other users and technologies (for an example, see section 2.2.2). A mobile network node has to be able to deal with these variations and dynamics in its sensory inputs in order to be able to successfully navigate a wireless network.

By definition wireless networks consist of more than one node, which means that they naturally lead to multi-robot scenarios when mobile nodes are used. Therefore, the robots have to be able to, besides navigating in a wireless network, interact with each other in a meaningful way. This implies tasks ranging from as simple ones as maintaining a network connection between two mobile robots through to the complete deployment and ongoing optimization of a wireless multihop network consisting only of mobile nodes. These kinds of algorithms can be designed either using a central controller or self-organized behavior.

In the context of a wireless network, a central controller however is impractical because of the associated communication costs. The control messages used to instruct the individual nodes reduce the overall bandwidth of the wireless network, which can go as far as preventing communication entirely. In principle these messages could be transmitted via another communication channel but that channel could as well be used for communication so that does not change the initial argument. Therefore, i.e., because of efficiency and scaling reasons, self-organized algorithms are preferred.

This adds a swarm robotics aspect to network robotics. Swarm robotic systems in general are often easier to implement and less computationally costly than a centralized solution and are potentially more robust. Algorithms for swarm robotics systems often work by loosely coupling the individual robots via their sensorimotor loops without using direct communication and are therefore ideally suited for a network robotics context. This also means that such algorithms only use locally available information.

While swarm robotics algorithms are usually more scalable, and in the context of wireless networks also more efficient in terms of bandwidth usage, they are generally speaking more challenging to design. The main reason for this is that they rely on emergence to generate global behavior from the local behaviors of the individual robots and the global behavior is not directly encoded in the local control strategies of individual robots. There exist known classes of global behavior such as for example foraging or pattern formation, for which the corresponding local controllers leading to this global behavior are known but there exist no generalized design patterns for arbitrary global behavior. This also means that swarm robotics algorithms are not easily grasped by analytic tools to for example be able to prove certain guarantees for a given behavior. This would be especially important for the fields of safety and security of such systems. These issues have to be addressed when designing

algorithms for network robotics with multiple robots.

In summary, network robotics is an interesting and promising field for robotics research. It combines aspects from sensing and navigation tasks in wireless networks, which show challenging nontrivial dynamics, with complex swarm robotic aspects as network robotics often implies multiple — potentially a large numbers of — robots interacting with each other. This field is also gaining more and more technical relevance for applications as (ad hoc) wireless networks become ever more prevalent.

1.2. Objective

From a pragmatic point of view, a problem statement for this work can be formulated, which includes all the interesting aspects of network robotics and the corresponding attractive research question from a robotics point of view. The general problem statement for this work can be summarized as:

> Investigate control strategies for autonomous mobile nodes of a wireless ad hoc network enabling them to work in a self-organized fashion, which only relies on local information.

In this context, using only local information means for single robots, to (if possible) not depend on globally referenced measurements such as from a Global Positioning System (GPS) system, but also for multiple robots to design scalable algorithms for self-organization based on emergence as opposed to centralized ones. These two requirements fit well with bio-inspired algorithms and can also potentially lead to more robust, flexible and scalable algorithms.

As such, this problem statement can readily be decomposed into the two main parts dealing with navigation in wireless networks with single robots on the one hand and with self-organization of multiple robots on the other hand. Additionally, real world wireless measurements as well as literature reviews on the state-of-the-art in both areas are necessary.

1.3. Approach

The approach adopted in this work follows several very general themes. These are derived from the problem statement as stated in section 1.2 and also result from the scientific focus and interest of the author. They should not be seen as a rigid structure to be followed but more like guidelines for a general orientation of this work.

The need for embodied experimentation is a common thread throughout this work. This need is based on the so-called embodiment hypothesis, which states that cognition (in natural as well as artificial systems) is shaped by the body interacting with the real world. This means on the one hand that cognition cannot be separated from the body, i.e., sensory input, morphology and computational paradigms and capabilities, but on the other hand also that algorithms intended to be used in the real world can only be designed by taking into account a real world bodily integration. This implies that they have to at least be evaluated on a real

robot. In their book „How the body shapes the way we think“, Rolf Peifer and Josh Bongard (Pfeifer and Bongard, 2006) give a more in depth explanation of the concept and underline it with a large number of examples. In principle, this concept of embodiment is a holistic one. However, this work does not focus on the morphology of the robot, in the sense that only very simple morphologies with low degrees of freedom like ground-based wheeled robots are used. Instead it focuses on sensory interaction with the real world. Nevertheless, the morphology of the robot is an important aspect, which cannot be dismissed and has to be kept in mind when designing algorithms. In the context of this work, focusing on the interaction with the real world, i.e., interaction with real wireless networks, also implies extensive measurements of real world dynamics backed by physical theory of electromagnetic waves.

There are many natural, mainly biological, systems interacting very successfully with the real world. Therefore, a very general approach is to use these systems as a starting point for developing technical systems. This approach is called bio-inspiration and there are many successful examples following this approach (Pfeifer et al., 2007). For this work, bio-inspired methods are especially interesting for the areas of chemotaxis and swarming. These bio-inspired solutions are already very good in terms of performance but implementing them on real robots also furthers the understanding of underlying principles implemented in biological systems.

Besides purely bio-inspired algorithms for taxis, for example chemotaxis by bacteria, it is also worthwhile to take into account other methods of navigation in scalar fields. One very important class of algorithms is based on gradients, which exist in a number of other fields such as optimization and learning theory. Therefore there exists a large body of theoretical work in this field called stochastic optimization. Analytic results from this field can be used to understand general convergence criteria and the behavior of these algorithms.

For more complex tasks, which cannot easily be solved only using reactive behavior, a very successful approach is the use of internal models (see chapter 3). The origins of this concept are two-fold, it emerged simultaneously in biology, specifically neuroscience, and in control theory, for example in systems with time delays to anticipate necessary future motor commands. Ordinarily, an internal model consists of a forward model (predictor) and an inverse model (controller). In this work, the forward model, i.e., the predictor part, will be used most frequently because the controller can often be formulated in a simple way for most of the systems studied here; heuristically or using a bio-inspired algorithm. The forward model can in general either be learned or be based on prior system knowledge. Both options will be explored. An extension of the often purely body-centric forward models are internal simulations, which imply a more long-term prediction into the future and often also include the environment and other agents.

These different themes are obviously very interconnected and also connect with a lot of other topics not mentioned in this brief discussion. In my opinion it is therefore very important to have a holistic view on the whole problem even when focusing on small details to be solved. This point of view is very closely connected to the embodiment hypothesis.

1.4. Structure of the Work

This work is split into three major parts: in the first part fundamentals dealing with basic concepts used throughout this work are discussed. In the second part on network robotics with single robots, measurements as well as algorithms for navigation of single robots in wireless networks are presented. The third part on self-organization with multiple robots finally discusses challenges and open questions as well as a possible solution for self-organization of multiple robots in the network robotics context. Properties and an implementation of this solution are presented.

In part I fundamentals on wireless communication and internal models are discussed. This part focuses on two topics, which are are of key interest for this work, and therefore discussed in detail separately. First, chapter 2 discusses the physical and technical foundation of wireless communication, focusing on analytical results but also presenting some measurements performed in real wireless networks. This chapter is to provide a clearer understanding of the basics of wireless communication, for example radio wave propagation, spectrum allocation, or antennas, which are key concepts for part II. Then chapter 3 gives an introduction into and an overview of the approach of internal models, which is a key concept that is used extensively in part III but also to a lesser degree in part II. Moreover, this concept has influenced most of the work presented here, even if it was not used explicitly. The historical development of the concept in the fields of neuroscience/biology as well as control theory are covered. Additionally, the question of grounding these internal models is presented briefly. Furthermore, machine learning methods, employed in this work as representations for internal models, are introduced briefly.

Algorithms for single robots (mainly) navigating in wireless networks are discussed in part II. An introduction into the topic of network robotics as well as a literature review of the state-of-the-art in network robotics is given in chapter 4. Chapter 5 presents the results of a number of different real world measurements of wireless network parameters. Then a robust algorithm for large-scale network exploration is presented in chapter 6. Chapter 7 and chapter 8 then introduce two algorithms for navigation in wireless networks. The first one is based on gradients and convergence of this algorithm is shown for the special case of measuring signal strength in a wireless network. The second algorithm is based on internal models and learns this model to predict signal strength at places not yet visited. Both algorithms can be extended to perform a wide number of tasks besides pure navigation, for example bridging two network nodes.

Self-organization of multiple robots in the context of wireless networks is discussed in Part III. A literature review on the state-of-the-art for self-organization, specifically in the network context, is given in chapter 9. In this part, reasons for why those algorithms are not yet used in real world scenarios and solutions to these issues are discussed. In chapter 10, an internal model based architecture proposed in the literature is introduced, which aims to resolve these issues. A real experimental implementation of this architecture as well as some modifications, following from real world constraints such as limited computational capabilities, are discussed. This architecture is then employed to implement a minimally ethical robot in chapter 11, which is cognizant of the consequences of its possible future next actions. This experiment is on the one hand a demonstration of the effectiveness of

the architecture and on the other hand also shows how a robot can become cognizant of the consequences of its own actions not only for itself but also for other robots with which it is interacting. Chapter 12 further demonstrates how this architecture can implement safety in multi-robot scenarios by using the knowledge of the consequences of interactions between multiple robots. Finally, further experiments dealing with aspects of learning and validating internal models and/or the respective observed real behavior against each other in the context of multi-robot scenarios are discussed in chapter 13.

In part IV the work is concluded. Specifically, section 14.1 presents the summary and section 14.2 the discussion of the work. In section 14.3 possible directions for future work are shown.

Finally, appendix A presents work performed specifically on flying robots as an alternative to the ground-based robots used in most experiments and appendix B presents work on small flying robots for safer indoor experimentation.

1.5. Publications

This work resulted in several publications. They are listed here as well as detailed analysis of the contributions of the different authors for transparency. In accordance with the doctorate regulations (Promotionsordnung) of the Mathematisch-Naturwissenschaftliche Fakultät at the Humboldt-Universität zu Berlin, Section 6, Article 2.b, this thesis is based on the works presented in the following articles:

Conference Proceedings

- „An Autonomous Flying Robot for Network Robotics“ by Christian Blum and Verena V. Hafner (Blum and Hafner, 2012): Christian Blum performed the experiments and analysis and wrote the bulk of the text. Robert Sombrutzki supported in setting up the Humboldt Wireless LAB (HWL) side of the experiment and Verena V. Hafner had an advisory role and supported writing the text.

- „Robust Exploration Strategies for a Robot exploring a Wireless Network“ by Christian Blum and Verena V. Hafner (Blum and Hafner, 2013): Most of the work done on this paper was performed by Christian Blum. Verena V. Hafner had an advisory role and supported writing the text.

- „Intuitive Control of Small Flying Robots“ by Christian Blum, Oswald Berthold, Philipp Rhan, and Verena V. Hafner (Blum et al., 2014): This work was performed in the context of the Diploma thesis of Philipp Rhan who was supervised by Christian Blum. Philipp Rhan reverse engineered the protocol of the Tracking system, implemented the software system, and performed the experiments. Christian Blum designed and implemented the interface between the Arduino and the remote control for the robot, helped set up the tracking system, and helped tuning the controllers. The creation of the text was a joint effort by the authors led by Christian Blum.

- „Towards an Ethical Robot: Internal Models, Consequences and Ethical Action Selection“ by Alan F. T. Winfield, Christian Blum, and Wenguo Liu (Winfield et al., 2014): This work is based on the original idea by Alan F. T. Winfield for the concept of the Consequence Engine (CE) (Winfield, 2014) as well as the hypothetical experiment proposed in that paper. The experiment as well as the analysis was implemented and performed by Christian Blum. Wenguo Liu assisted with some of the hardware aspects of the experiment. The creation of the text was a joint effort of Alan F. T. Winfield and Christian Blum.

arXiv.org

- „Gradient-based Taxis Algorithms for Network Robotics“ by Christian Blum and Verena V. Hafner (Blum and Hafner, 2014): Most of the work done on this paper was performed by Christian Blum. Verena V. Hafner had an advisory role and supported writing the text.

Journal Papers in Preparation

- „Internal model based safety“ by Christian Blum, Alan F. T. Winfield, and Verena V. Hafner (Blum et al., 2015): This work is based on the original idea by Alan F. T. Winfield for the concept of the CE (Winfield, 2014) and the work presented in (Winfield et al., 2014). The experiment as well as the analysis was implemented and performed by Christian Blum. The creation of the text was a joint effort of all authors.

Conference Papers in Preparation

- „Active exploration of sensor networks from a robotics perspective“ by Christian Blum and Verena V. Hafner (Blum and Hafner, 2015): Most of the work done on this paper was performed by Christian Blum. Verena V. Hafner had an advisory role and supported writing the text.

Conference Posters not in Proceedings

- „Tactile Sensors for Learning of Soft Landing on a Flying Robot“ by Jan Gosmann, Christian Blum, Oswald Berthold, and Verena V. Hafner (Gosmann et al., 2013): The experimental work and analysis for this paper was performed in the context of the lab rotation of and by Jan Gossmann. Christian Blum advised on the general design, helped interfacing the sensors used to an Arduino and performed some test flights. Oswald Berthold also helped with hardware issues and performed some test lights. The creation of the text was a joint effort of all the authors led by Jan Gossmann.

Part I

Fundamentals

Chapter 2

Wireless Communications Basics

As stated in section 1.1, network robotics, which deals with mobile autonomous wirelessly connected network nodes, is an attractive field of research for robotics researchers with growing practical relevance. IEEE 802.11 Wireless Local Area Network (WLAN) was chosen to be used as a wireless technology because it operates in unlicensed Industrial, Scientific and Medical (ISM) bands (see section 2.2.2). This means that, in contrast to for example cellular networks operating in a licensed band, network parameters, choice of protocol, and so on can be freely varied in a wide range, which allows for more flexibility in experimentation. This however also means that there are potentially other users using the same ISM band, which can lead to interference between different users and/or communication technologies. Furthermore, WLAN is often available on robots and/or laptops and can also be added easily via cheap USB adapters. Because of these reasons, WLAN is also often the communication technology of choice for wireless multihop networks. In this chapter, the physical and technological foundations of wireless communication with a specific focus on WLAN are introduced.

2.1. Maxwell's Equations

Maxwell's Equations describe the properties of electric and magnetic fields and their dynamics (Jackson, 1962). In the presence of media, they take on this form:

$$\boldsymbol{\nabla} \cdot \mathbf{D} = \rho_\mathrm{f}, \tag{2.1.1a}$$

$$\boldsymbol{\nabla} \times \mathbf{E} = -\partial_t \mathbf{B}, \tag{2.1.1b}$$

$$\boldsymbol{\nabla} \cdot \mathbf{B} = 0, \tag{2.1.1c}$$

$$\boldsymbol{\nabla} \times \mathbf{H} = \mathbf{j}_\mathrm{f} + \partial_t \mathbf{D}. \tag{2.1.1d}$$

Here, $\mathbf{E}$ and $\mathbf{H}$ are the electric and magnetic field, $\mathbf{D}$ and $\mathbf{B}$ are the electric and magnetic flux density and ρ_f and $\mathbf{j}_\mathrm{f}$ are the free charge and current density, respectively. These four equations form the base of the present work.

The electric field $\mathbf{E}$ and the magnetic flux density $\mathbf{B}$ are connected to the electric flux density $\mathbf{D}$ and the magnetic field $\mathbf{H}$ via the polarization $\mathbf{P}$ and the magnetization $\mathbf{M}$, respectively. They are defined as

$$\mathbf{D} = \varepsilon_0 \mathbf{E} + \mathbf{P}, \tag{2.1.2}$$

$$\mathbf{H} = \frac{1}{\mu_0}\mathbf{B} - \mathbf{M}. \tag{2.1.3}$$

Both the polarization and the magnetization depend on the properties of the medium. They are both zero in vacuum.

2.1.1. Free Space Wave Equation and Solution

In the absence of media the polarization and the magnetization are zero, the electric and magnetic flux density read

$$\mathbf{D} = \varepsilon_0 \mathbf{E}, \tag{2.1.4}$$

$$\mathbf{B} = \mu_0 \mathbf{H}. \tag{2.1.5}$$

Assuming furthermore no free charges or currents, i. e., $\rho_\mathrm{f} = 0$ and $\mathbf{j}_\mathrm{f} = 0$, Maxwell's Equations become

$$\boldsymbol{\nabla} \cdot \mathbf{E} = 0, \tag{2.1.6a}$$

$$\boldsymbol{\nabla} \times \mathbf{E} = -\partial_t \mathbf{B}, \tag{2.1.6b}$$

$$\boldsymbol{\nabla} \cdot \mathbf{B} = 0, \tag{2.1.6c}$$

$$\boldsymbol{\nabla} \times \mathbf{B} = \mu_0 \varepsilon_0 \partial_t \mathbf{E}. \tag{2.1.6d}$$

Now the curl operator $\boldsymbol{\nabla}\times$ can be applied on eq. (2.1.6b) and eq. (2.1.6d) differentiated in time. Plugging one into the other and making use of eq. (2.1.6a) and eq. (2.1.6c) yields the wave equation

$$-\boldsymbol{\nabla} \times \boldsymbol{\nabla} \times \mathbf{E} = \frac{1}{c_0^2}\partial_t^2 \mathbf{E}, \tag{2.1.7}$$

where $1/c_0^2 = \mu_0 \varepsilon_0$ is the vacuum speed of light. The solutions to the wave equation are plane waves:

$$\mathbf{E}(\mathbf{r}, t) = \mathbf{E}_0 e^{\mathrm{i}(\mathbf{k}\mathbf{x} - \omega t)}, \tag{2.1.8}$$

where $\mathbf{k}$ is the wave vector, which is related to the wavelength via $\lambda \left|\mathbf{k}\right| = 2\pi$ and ω is the angular frequency. In free space they are related by $\left|\mathbf{k}\right| \omega = c_0$. Due to Maxwell's Equations, the vectors $\mathbf{E}$, $\mathbf{B}$ and $\mathbf{k}$ have to obey

$$\mathbf{B} = \frac{1}{c_0 \left|\mathbf{k}\right|}\mathbf{k} \times \mathbf{E}, \tag{2.1.9}$$

which is shown schematically in fig. 2.1.

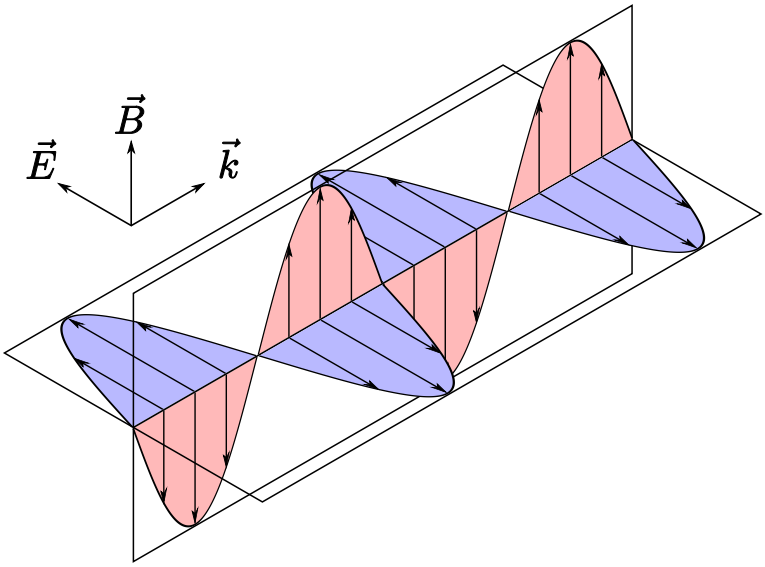


Figure 2.1.: A plane wave in free space

2.1.2. Electromagnetic Waves in Media

In general, analytic solutions to Maxwell's Equations only exist for special simplistic cases in media, like infinite purely dielectric media, because the polarization $\mathbf{P}$ and magnetization $\mathbf{M}$ in eq. (2.1.2) depend on the exact distribution of the different media. This already hints at the difficulties of accurately simulating the propagation of electromagnetic waves in that a simulation can only be as accurate as the knowledge about the distribution of the media, i.e., about the environment. For a real world scenario, moving a desk or opening or closing a door can already have a drastic impact. Thus, in this section only high-level mechanisms will be discussed.

In media and at interfaces between different media, three basic effects occur: reflection, scattering and diffraction (Rappaport, 2001). Usually, these effects are combined into effective propagation models, which will be discussed in detail in section 2.3.

Whenever an electromagnetic wave crosses an interface between two media with different material properties, it is partially reflected and partially transmitted. In the special case of an electric conductor as a second medium, the wave is reflected completely. Reflections usually also change the polarization of the wave.

Interaction of an electromagnetic wave with particles smaller and of the order of the wavelength of the electromagnetic wave is called scattering. The wave is diffused, i.e., spread out by diffraction. For larger objects, the surfaces can often be modeled as reflective surfaces instead.

Diffraction occurs if there is an object with a sharp edge or other sharp features in the path of an electromagnetic wave. According to the Huygens-Fresnel principle (Jackson, 1962), which states that every point on the wavefront of an electromagnetic wave can be considered the source of a secondary spherical wave. Interference between all these waves then leads to

a new wavefront and so on. Diffraction is then caused by the propagation of the secondary waves originating at and next to the edge of the object and interfering in the shadowed area behind the object.

2.1.3. Simulation Methods

One of the most straight-forward methods for the simulation of electromagnetic waves, i.e., to solve Maxwell's Equations numerically is to use finite differences. This effectively discretizes Maxwell's Equations on a regular grid and leads to a number of finite difference equations from which a leapfrog integration scheme can be derived.

The modern version of this method, known as finite-difference time-domain (FDTD), was first described in 1966 by Yee (1966). The fundamental idea behind his formulation of the algorithm is to interleave the electric and magnetic fields in space and time using centered finite difference operators for enhanced precision. This method has been improved continuously throughout the years including significant improvements in the areas of grid truncation techniques and sub-pixel smoothing as well as efficient implementation (Taflove and Hagness, 2005).

One very popular free and open source implementation of FDTD is MIT Electromagnetic Equation Propagation (MEEP) (Oskooi et al., 2010). There are numerous other commercial as well as free implementations for CPUs as well as GPUs.

The discretization grids of FDTD have to be able to resolve the details of the material distribution and also as the wavelength of the electromagnetic waves. In general, a minimum resolution of 8 pixel per wavelength is necessary as a result of the concrete interleaved discretization scheme. Time discretization follows automatically from the spatial discretization via the Courant factor S (Courant et al., 1928), which is 0.5 in MEEP per default, via $\delta t = S\delta x$.

As an example, consider the discretization of an area for the simulation of $2.4\,\mathrm{GHz}$ radio waves in two dimensions. If an area of $10\,\mathrm{m} \times 10\,\mathrm{m} \times 2\,\mathrm{m}$ of a small office environment is to be simulated and a resolution of 10 pixel per wavelength is chosen, this leads to a discretization grid of $1.25\,\mathrm{cm}$ spacing with $102,400,000$ pixels. Typically $144\,\mathrm{B}$ per pixel are needed to model the fields and material (Davidson, 2005). This leads to a memory consumption of $14.75\,\mathrm{GB}$. Additionally, a description of all the material in the simulated area is needed on the $1.25\,\mathrm{cm}$ grid. This means the complete knowledge of the physical parameters of the environment for all grid points. In a real world scenario this would include walls, wall composition, furniture, electric equipment, electric cables, persons, and so on. Just attaining this data is a task difficult to solve in a real world scenario[1]. Run time is mainly affected by the used processor and available memory bandwidth. The simulation is usually memory bandwidth limited because the FDTD scheme leads to only few arithmetic instructions per memory load instruction.

As can be seen from this example, FDTD can become very costly, especially in 3D. Thus

[1] A big part of the work done in the COST-231 project (Damosso, 1998; Damosso and Correia, 1999) was actually to perform electromagnetic sounding of different types of commonly found walls, corridors and general indoor and outdoor scenarios. As expected, huge variations in the propagation characteristics were found depending on the materials and configurations.

in the area of wireless communication technologies it is mainly used for items such as antenna design or the modeling of specific absorption rate (SAR). For the actual calculation of propagation characteristics, for example inside of buildings to calculate coverage areas, etc., usually less accurate techniques are used. One prominent method for these kinds of calculations is ray-tracing which is a lot more efficient than FDTD. However, if interference and small scale fading is important, it is not accurate enough because it cannot model near field effects such as diffraction. Additionally, a very good description of the environment is needed also for this technique.

2.2. Technical Aspects of Radio Waves

2.2.1. Basic Network Protocol Characteristics

The Open Systems Interconnection (OSI) model (Tanenbaum and Wetherall, 2012) is a layered reference model for network communication. It has been standardized both by the International Telecommunication Union (ITU) and International Organization for Standardization (ISO) and has been widely used for the design of network protocols.

In the context of this work the two lowest layers are the most relevant. The first layer, the Physical Layer, defines electrical and physical specifications as well as the corresponding protocols necessary for data connection. In the case of wireless communication this includes items such as electromagnetic spectrum allocation or modulation. The second layer, the Data Link Layer, provides data transfer between nodes and also defines error correction methods. It is split into two sublayers: Logical Link Control (LLC) and Media Access Control (MAC). The LLC layer is responsible for multiplexing protocols when transferring data over the MAC layer as well as for error correction and node-to-node flow control. The MAC layer provides channel access control mechanisms and addressing, to enable communication within a shared communication medium, such as the air in the case of wireless communication. For the case of WLAN, which is mainly used in this thesis, these two first layers are specified by the `IEEE 802.11` specifications.

The MAC layer thus specified unique identifiers for all network interfaces (of which a logical network node can have several). This means in the context of this work that measurements of network parameters for instance Received Signal Strength Indication (RSSI) can be uniquely attributed to the corresponding network interfaces and thus also to the corresponding network nodes. Therefore, even though the communication medium is shared, measurements corresponding to specific network nodes can uniquely be identified.

2.2.2. Spectrum

The electromagnetic spectrum is divided into frequency bands for different uses according to the frequency allocation scheme decided on by international and national regulatory organizations such as the International Telecommunication Union (ITU). Because of the wide distribution and availability, WLAN (based on `IEEE 802.11`) (Group et al., 2010), was chosen as a radio communication technology for this work. According to specification, it uses the Industrial, Scientific and Medical (ISM) bands in the $2.4\,\mathrm{GHz}$ and $5.8\,\mathrm{GHz}$ range.

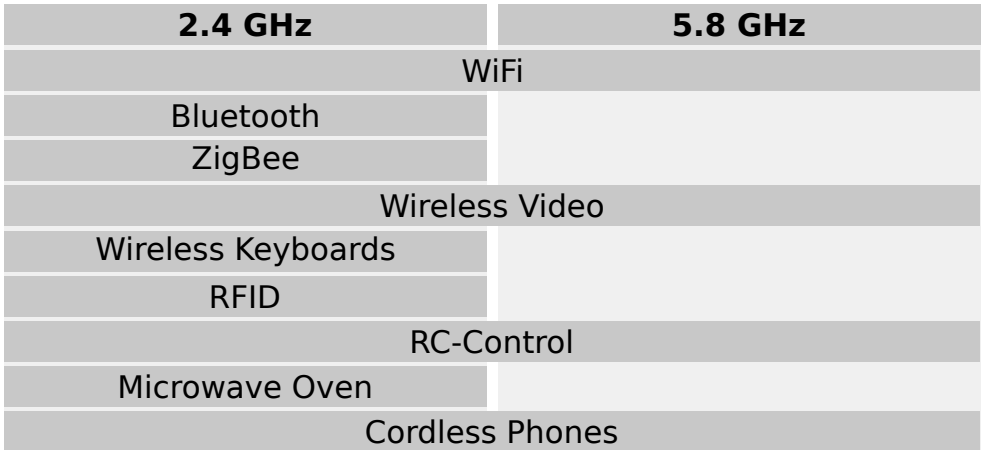


Figure 2.2.: Usage examples of the two ISM bands at $2.4\,\mathrm{GHz}$ and $5.8\,\mathrm{GHz}$

As depicted in fig. 2.2, there are other technologies besides WLAN also operating in these frequency ranges, which potentially means a lot of interference in these frequency bands. For most potential interference candidates in practice this means additional noise during communication. Usually, the underlying radio technology of WLAN is able to cope with this noise but in extreme cases this can lead to complete jamming of certain channels, denying communication. In practice, such extreme cases are very rare.

To illustrate this situation for a typical office environment an experiment was conducted. Using a Texas Instruments CC2500 low-power $2.4\,\mathrm{GHz}$ RF transceiver, which supports 256 channels in the $2.4\,\mathrm{GHz}$ band in the range of $2400\,\mathrm{MHz}$ to $2483.5\,\mathrm{MHz}$, the $2.4\,\mathrm{GHz}$ band was monitored in a typical office environment for over four days. These 256 bands were sampled randomly with a rate of $25\,\mathrm{kHz}$ measuring the RSSI. For these samples the number of samples with an RSSI over $-95\,\mathrm{dBm}$, i.e., samples where the received energy was higher than the (mostly thermal and read-out) background noise, were counted for the respective channels. The result is depicted in fig. 2.3.

The measurements show the typical broad signatures of wireless LAN signals as well as several other signatures. The two broad signatures extending over the complete measurement period can be traced back to eduroam and some university WLANs operating in the channels 1, 9, and 13 respectively. Narrow-band signals extending over hours are probably wireless keyboards, mice or audio devices. Bluetooth devices cannot be easily seen in this kind of plot because they employ frequency-hopping strategies. Signals covering the whole bandwidth of the $2.4\,\mathrm{GHz}$ band can be traced back to remote controls for radio controlled vehicles in the lab. The day and night cycle are easily identified and workdays distinguished from weekends by the amount of traffic during day time. The distinguished short bursts of wireless LAN activity in channel 1 during Thursday night are due to experiments in the wireless mesh network testbed HWL which were known to be scheduled for that night. It is clear from this experiment that in a real world scenario there is a multitude of interference sources for WLAN with time-varying characteristics.

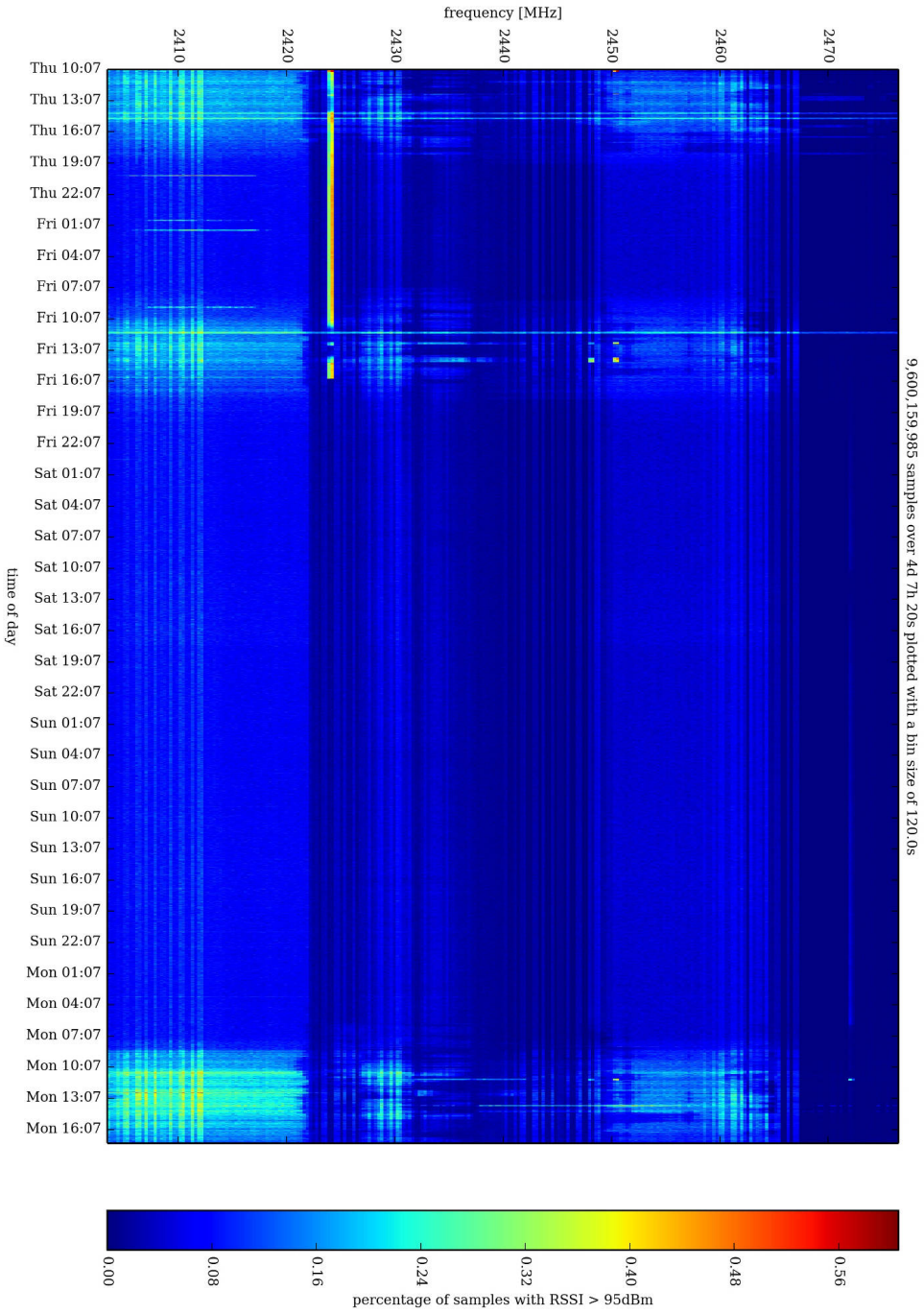


Figure 2.3.: Measurement of 2.4 GHz ISM band over four days in an office environment

2.2.3. Antennas

A radio transmitter generates electrical oscillations at the radio frequency and feeds this signal to an antenna which in turn converts this signal into electromagnetic waves (and vice versa for a radio receiver). Depending on the shape and size of the antenna, this conversion is more or less efficient for different frequencies and directions. Three different example antennas and their radiation patterns are depicted in fig. 2.4. Isotropic antennas are only used for theoretical calculations as a reference antenna and do not exist as real antennas as they are point-like.

Apart from their radiation patterns, antennas are characterized by their respective gain, which is defined as the output in the direction of maximum output in relation to the output of an isotropic radiator for the same feed power and is measured in dBi. Almost all antennas used for mobile applications are omidirectional, meaning that the azimutal radiation pattern is circular. Practically, either rod antennas or printed circuit board (PCB) antennas (STMicroelectronics, 2011) are widely used for the reason of omnidirectionality.

All radiation emitted by any antenna is polarized. For most antennas, including rod antennas, the polarization of the emitted electromagnetic waves is linear but there also exist antennas emitting circular polarization. This means if the polarization of the emitting antenna is not the same as the receiving antenna, there are losses. If both antennas emit linear polarized electromagnetic waves and are arranged perpendicular, the receiving antenna will not pick up any signal at all. If a circular polarized signal is received by a linear antenna, half of the signal strength, i.e., about $3\,\mathrm{dB}$, is lost. Reflections also affect polarization, which can become relevant for multi-path interference. In practice this means that using omnidirectional linear antennas and since these are usually almost parallel (perpendicular to the ground), there are almost no losses. Nevertheless it is important to know that huge losses can occur if one is not careful about antenna orientation.

2.3. Radio Wave Propagation Models

The description of radio wave propagation models in this section loosely follows Goldsmith (2005) and Rappaport (2001). Radio wave propagation models can roughly be divided into large scale and small scale models. Large scale models model propagation for distances of the order of several hundred or thousand wavelengths leading to so-called path loss models, while small scale models model propagation for distances on the order of a few wavelengths where interference effects are important and which are known as small scale fading models. The next two sections discuss both classes of models in more detail.

Figure 2.5 shows an example measurement of RSSI for an arbitrary line in an office environment. The measurements were conducted using an robosoft RobuLAB 10 using regular IEEE 802.11g wireless LAN. The measurement shows both kinds of variations. There are small scale variations of the order of the wavelength of the wireless signal($12.5\,\mathrm{cm}$), however, as can be seen in the filtered data, there is also a slowly varying variation, which can be identified by large-scale path loss.

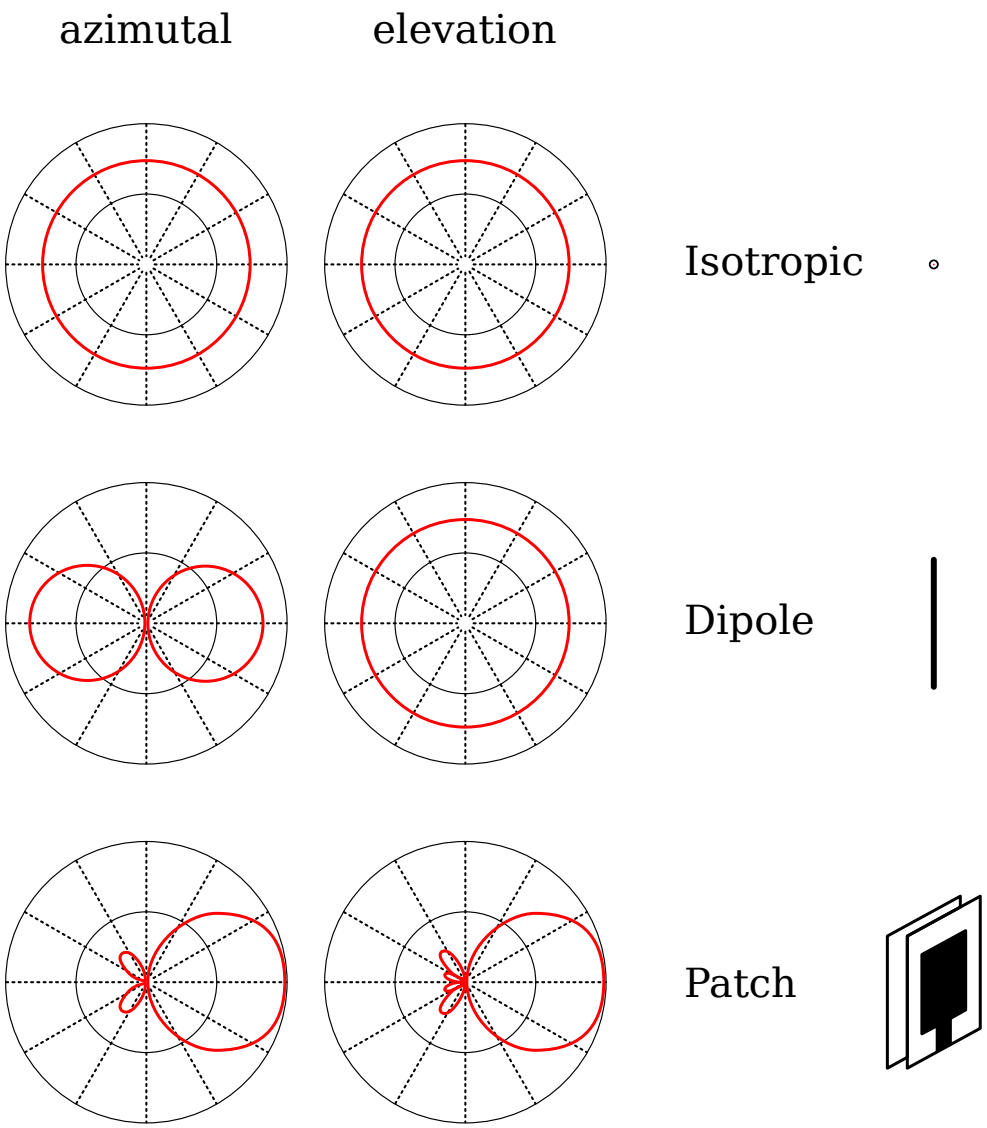


Figure 2.4.: Radiation patterns for three different antennas

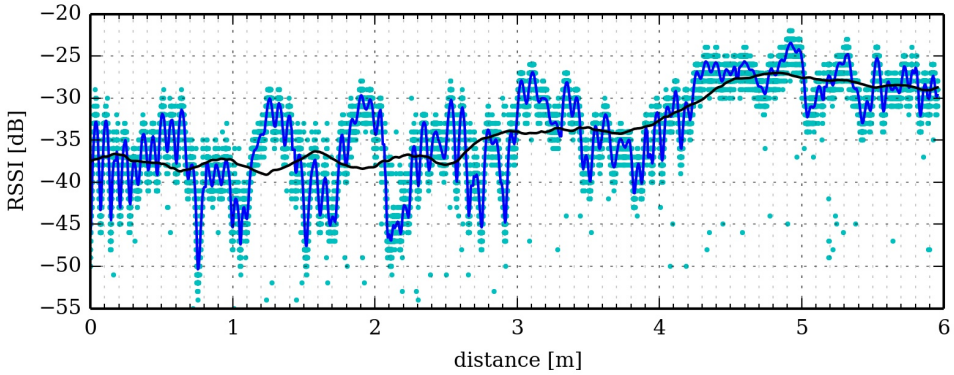


Figure 2.5.: RSSI measurements with RobuLAB 10

2.3.1. Large Scale Path Loss

Path loss P_L at the distance d between transmitter and receiver is defined as the ratio between transmitted power P_t and received power P_r as:

$$P_L(d) = \frac{P_t}{P_r}. \tag{2.3.1}$$

Usually logarithmic units are used for P_L:

$$P_L(d)[\mathrm{dB}] = 10 \log_{10} \frac{P_t}{P_r} \tag{2.3.2}$$

In free space, the received power is given by

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \tag{2.3.3}$$

where G_t and G_r are the transmitter and receiver gain, L the system loss, i.e., loss in the amplifier etc., and λ is the wavelength of the radio wave. Usually we assume $L = 1$. This formula is valid for $d > d_f$ where d_f specifies the Fraunhofer region ($d_f >>$ antenna size and $d_f >> \lambda$). For an isotropic radiator this formula can be understood intuitively since the radiated power is spread over a sphere of radius d and the surface of this sphere scales with d^2 so the radiated power for a specific direction decreases with d^{-2}. In logarithmic units this becomes

$$P_L(d)[\mathrm{dB}] = P_L(d_0) + 10 \log_{10} \frac{d}{d_0} \tag{2.3.4}$$

with the reference path loss $P_d(d_0)$ at a reference distance d_0 (usually $1\,\mathrm{m}$). This reference path loss is used to absorb all the constants like antenna gains into a single number.

For more general scenarios, where other media exists, i.e., walls, buildings, furniture, etc., every general scenario and wavelength leads to a different empirical path loss model. Usually these empirical path loss models are generated from data measured in a specific situation,

Building	γ
Retail Store	2.2
Grocery Store	1.8
Office, hard partition	3
Office, soft partition	2.6
Office, multiple floors	$2-6$
Factory	$1.6-3.3$
Home	3

Table 2.1.: Empirical γ values for the simplified path loss model, taken from Goldsmith (2005) and Rappaport (2001)

for example large urban cells at around 1 GHz in the COST-231 project (Damosso, 1998; Damosso and Correia, 1999).

Indoor path loss can often be approximated by a distance power law for the received power

$$P_r(d) \propto \frac{1}{d^\gamma} \tag{2.3.5}$$

where γ is an empirical factor depending on the environment. A γ value of 2 yields the standard free space model. Typical values are shown in table 2.1.

In logarithmic units this becomes

$$P_L(d)[\mathrm{dB}] = P_L(d_0) + 10\gamma \log_{10} \frac{d}{d_0}. \tag{2.3.6}$$

For indoor scenarios, indoor attenuation factors are often used in addition to the empirical distance power law discussed above. These are measured loss factors for different objects such as walls or doors. They are incorporated into the path loss as

$$P_L(d)[\mathrm{dB}] = P_L(d_0) + 10\gamma \log_{10} \frac{d}{d_0} + \sum_i AF_i \tag{2.3.7}$$

where the AF_i are the attenuation factors. table 2.2 lists some common indoor attenuation factors.

In real life, no two walls yield exactly the same attenuation factor and there is a lot of clutter in the environment such as assorted furniture etc. This means that even for the same environment, points with the same distance to the transmitter show different path loss. It is impossible to take all those variations for different path loss models into account. Real measured path loss models have shown to be statistical distributions following a log normal distribution, i.e., they show a normal distribution in logarithmic units. This effect is usually called log-normal shadowing in the literature. Thus, usually a random noise term is added to the empirical path loss function as well as to the attenuation factors to account for this variability [2].

[2]Note that even though this factor is random, for a correct model it has to be fixed for the same spatial configurations at least for the duration of one simulation

Material Type	$\boldsymbol{AF}$[dB]
Cloth partition	1.4
Double plasterboard wall	3.4
Light textile	$3-5$
Empty cardboard inventory boxes	$3-6$
Metal catwalk/stairs	5
Concrete block wall	$13-20$
Aluminium siding	20.4
All metal	26
One floor	13
Two floors	19
Three floors	24

Table 2.2.: Empirical indoor attenuation factors, taken from Goldsmith (2005) and Rappaport (2001)

2.3.2. Interference and Small Scale Fading

A pulse transmitted by a transmitter will be reflected, diffracted and scattered in the environment, generating multiple different paths on which it can reach the receiver. Thus, what the receiver records is not the single pulse that was transmitted, but a main pulse (if there is a line of sight path) and then a number of secondary pulses, one per propagation path, which can be infinite in the real world due to secondary reflections etc. This situation is depicted schematically in fig. 2.6.

This train of pulses can potentially also overlap, i.e., interfere with each other. This interference leads to variations of the signal strength on scales of the order of the wavelength

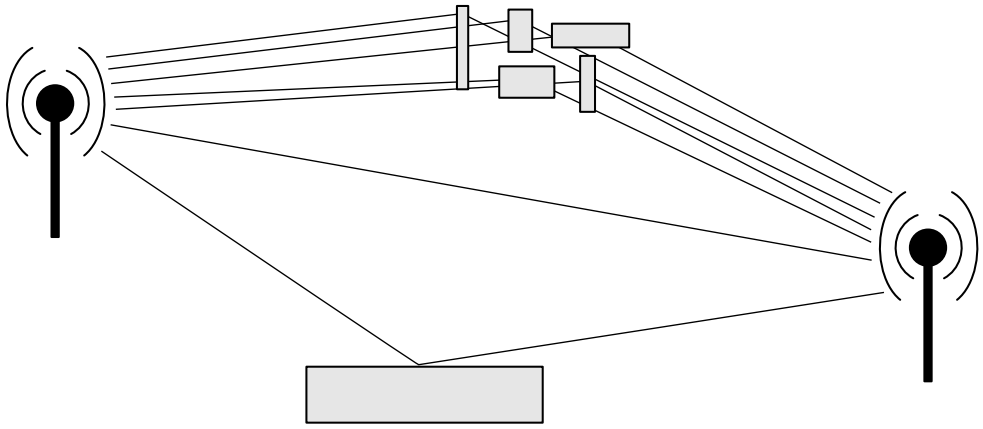


Figure 2.6.: Schematic diagram of different propagation paths contributing to the multi-path propagation between a transmitter and receiver.

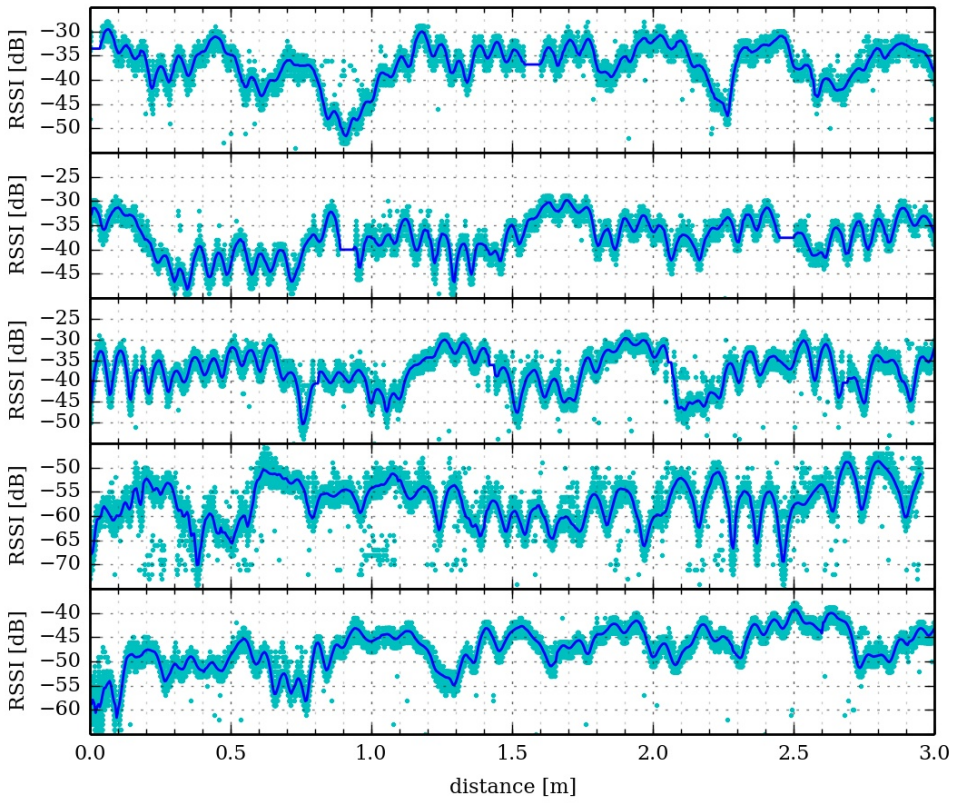


Figure 2.7.: RSSI measurements with robulab-10

of the radio wave, which is called small scale fading. Real example measurements showing small scale fading are depicted in fig. 2.7. The variations in signal strength are clearly of the order of the wavelength ($12.5\,\mathrm{cm}$ in this example).

A multi-path signal is also always time-varying in nature because either the transmitter or receiver move if they are mobile, or parts of the environment move such as for example people moving, doors opening and closing, etc. In fig. 2.7 this can be seen in the variations of the signal comparing the actual measurements against the moving average of the signal over $1\,\mathrm{cm}$. These variations are clearly larger than the variations due to the robot moving a distance of $1\,\mathrm{cm}$.

For the special case of a moving sender or receiver and direct line of sight conditions, this effect can be calculated analytically and is called Doppler shift. This effect can be experienced practically in the audible regime for a moving police car and the change in frequency of the siren when the police car passes by. Doppler shift leads to a frequency shift f_d, which for a far away source can be approximated by

$$f_d \approx \frac{v}{\lambda} \tag{2.3.8}$$

where v is the relative speed between transmitter and receiver and λ is the wavelength.

For $2.4\,\mathrm{GHz}$ and a speed of $10\,\mathrm{m\,s^{-1}}$ this yields a frequency shift of $f_d = 80\,\mathrm{Hz}$. IEEE 802.11 (wireless LAN) specifies $22\,\mathrm{MHz}$ wide channels with $5\,\mathrm{MHz}$ spacings. Thus the effects of Doppler shift are negligible in this range for normal speeds (Mahasukhon et al., 2007). This effect only becomes relevant for much higher speeds and/or more complex modulations such as employed by multiple-input and multiple-output (MIMO) techniques for example used in IEE 802.11n.

There are a lot of statistical models for small scale fading and multi-path effects depending on the frequency, modulation, environment, etc. Because of their statistical nature they are not that interesting in the use case of real robotics since we are looking for concrete instances of an environment in contrast to statistical properties of a class of environments. A full simulation of Maxwell's Equations could account for multi-path effects, which is computationally costly and requires full knowledge of the environment, which is impractical.

Chapter 3

Introduction to Internal Models

In this chapter the concept of internal models will be introduced in depth. This concept is used extensively in part III and chapter 8, but also influenced most of the work presented here as a way of thinking about intelligence. The concepts of grounded cognition, as well as machine learning methods used to learn and represent internal models, will be introduced as they are essential practical and philosophical aspects of the concept of internal models.

3.1. Grounding

Barsalou (2010, p. 717) defines grounded cognition as:

> According to classic theories, the core knowledge representations in cognition are amodal data structures processed independently of the brain's modal systems for perception, action, and introspection. From this perspective, the core representations in cognition differ from representations in modal systems, function according to different principles, and reside in a modular semantic system (Tulving, 1985). Grounded cognition is often defined negatively as the view that classic theories are incorrect: The core knowledge representations in cognition are not amodal data structures that exist independently of the brain's modal systems. Instead — according to a positive definition of grounded cognition — the environment, situations, the body, and simulations in the brain's modal systems ground the central representations in cognition. From this perspective, the cognitive system utilizes the environment and the body as external informational structures that complement internal representations. In turn, internal representations have a situated character, implemented via simulations in the brain's modal systems, making them well suited for interfacing with external structures.

This means that cognition is tightly bound to sensory and motor capabilities as well as the respective internal simulation mechanisms (Barsalou, 2008). Therefore, this theory is directly related to the concept of embodiment (Pfeifer and Bongard, 2006; Pfeifer et al., 2007) as well as the simulation theory of cognition as discussed in section 3.2.3.

In contrast to grounded cognition, the term embodied cognition is used to explicitly stress the importance of the sensorimotor interactions of the body with the physical world on

cognitive representations. Grounded cognition refers to cognition grounded in the physical properties of the world, which can but does not have to be encoded in bodily states. Unfortunately, both terms are often used interchangeably in the literature (Pezzulo et al., 2011).

An interesting overview on the connections between the fields of embodied cognition, metaphorical thought, and language, is given by Lakoff (2014). Lakoff presents a large number of examples of metaphors beginning with so-called primary metaphos, which are very close to embodied experiences, to complex abstract metaphorical blends. He then gives an overview of the current neural theory of metaphor and goes on to show how these neural theories could be connected to the different types of metaphors. He concludes stressing „the centrality of embodiment as *the* mechanism of meaninfulness “ (Lakoff, 2014, p. 12).

For an in-depth introduction into the topic with a discussion on the different grounded theories, empirical evidence as well as its issues and challenges refer to Barsalou (2008). Furthermore, Barsalou (2010) gives an historical overview of the topic, discusses current empirical evidence and proposes future research directions and challenges. The author also predicts that the different perspectives on cognition will converge in the future and presents grounds for this view.

Pezzulo et al. (2011) propose the use of robotics as a synthetic methodology to implement embodiment as a tool to advance theories of embodied cognition and enable testing. For this the authors believe that computational theories can have an important impact on embodied theories of cognition. In this context they discuss the requirements and potential impact of computational models (embodied in robots) on grounded theories. They also believe that this process can also provide insights into how to design better cognitive robotic systems.

Pezzulo (2011) proposes an evolutionary and developmental pathway which led from internal (forward) models for motor control to internal simulation and consequently knowledge representation, i.e., embodied cognition. The author explains in depth how different kinds of knowledge could be represented in this way (using internal models and internal simulation).

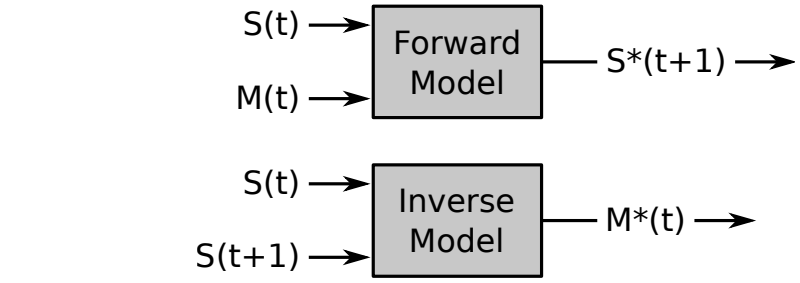
Internal models are thus one possible option to represent the knowledge about sensorimotor interactions.

3.2. Internal Models

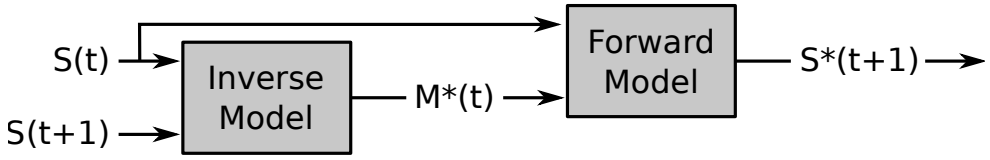
The idea of internal models can be retraced at least as far back as 1943 when K. J. W. Craik presented his idea of „small-scale models“ (Craik, 1967, p. 61):

> If the organism carries a „small-scale model“ of external reality and of its own possible actions within its head, it is able to try out various alternatives, conclude which is the best of them, react to future situations before they arise, utilize the knowledge of past events in dealing with the present and future, and in every way to react in a much fuller, safer, and more competent manner to the emergencies which face it.

This topic of internal models topic has gained traction especially since the 1990s in the fields of neuroscience, control theory and robotics, and is now an established concept in



(a) Forward and inverse models



(b) A pair of coupled forward-inverse models

Figure 3.1.: Forward and inverse models as well as a pair of coupled forward-inverse models. $S(t)$ and $M(t)$ denote the current sensory inputs and motor commands, $S(t+1)$ denotes the desired future sensory input, and $S^\star(t+1)$ and $M^\star(t)$ the predicted future sensory input and motor command respectively.

these fields.

3.2.1. Biological Origins

In biology and neuroscience, the concept of internal models is thought to be behind most of the exceptional sensorimotor skills that can be observed in nature, since it is essential for high-performance control as implemented in biological systems (Haruno et al., 2001; Wolpert et al., 2011).

Internal models are also believed to be behind self-agency. Weiss et al. (2011) present work on self-generated sound and Blakemore et al. (1998, 2000) present experiments on self-tickling supporting this hypothesis.

There are two main kinds of internal models: forward models (predictor), which predict future sensory inputs as a consequence of given motor inputs, and inverse models (controller), which supply motor commands leading to a desired sensory input(Wolpert et al., 1995). These two kinds of internal models are depicted schematically in fig. 3.1a. Additionally, there are models predicting physical properties of the environment (Flanagan et al., 2001; Zago et al., 2004). These models can also be coupled, and such a pair of coupled forward-inverse models is depicted schematically in fig. 3.1b. Together these models can be used to generate and test hypotheses about the outcome, i.e., consequences, of possible future actions and can also be used to recognize the behavior of other agents. In the long term, these systems could lead to a better understanding of the hypothesis of the theory of mind (Premack and Woodruff, 1978).

There is evidence in the form of corollary discharge (Sommer and Wurtz, 2008) in a lot

of invertebrates suggesting the use of internal models for sensorimotor skills (Webb, 2004). Recently, the first definitive use of internal models for interception steering has been shown for dragonflies (Mischiati et al., 2015). In vertebrates, especially in primates and humans, the hypothesis of the use of internal models is already well established. Examples range from internal physics models in humans for proprioceptive sensing (Merfeld et al., 1999) or forward and inverse models for grasping (Flanagan et al., 2001) to internal models for pitch control in human singers (Jones and Keough, 2008). In general, model-driven control, as implemented by the use of internal models for sensorimotor skills, overcomes the limitations of sensor delay by using the predictive power to generate motor commands for anticipated future sensor inputs, for example when trying to grasp a moving object.

Similar models in spatial navigation have been found to have a concrete neurological implementation in rats in the form of place-cells (O'Keefe, 1976) and head direction cells (Taube et al., 1990).

It was also proposed that the cerebellum implements such models (Wolpert and Kawato, 1998) and updates them to continuously adapt to new situation. This hypothesis is until now compatible with most experimental evidence (Ito, 2008) but a lot of experimental challenges in terms of neuroimaging remain unsolved (Stoodley, 2012).

Specifically, the idea of multiple paired inverse-forward model for motor control and trajectory-planning is prominent in neuroscience (Wolpert and Kawato, 1998; Kawato, 1999) and has been suggested to also facilitate these skills in humans (Haruno et al., 1999).

Because of the computational similarities between internal model driven sensorimotor skills and skills for social interaction, it has been suggested that the latter have been developed by extending the former (Wolpert et al., 2003). Even though this hypothesis remains controversial (Jacob and Jeannerod, 2005; Kilner, 2011), the recent discovery of the mirror system (Rizzolatti and Craighero, 2004) in primates (Rizzolatti et al., 1996) and subsequently in humans (Grèzes et al., 2003; Keysers and Gazzola, 2010), suggests that those internal models can at least be used for action recognition. A more recent review on this topic with a special focus on computational requirements was presented by Oztop et al. (2006). This hypothesized mechanism has also been implemented successfully in robotic systems (Demiris and Khadhouri, 2006).

3.2.2. Control Theory

In control theory, the use of internal models is well established in areas like model driven control (Garcia and Morari, 1982) or predictive control (Morari and Lee, 1999). In general the idea is that the use of coupled forward models (predictors) and inverse models (controllers) can overcome problems of more complex motor tasks posed by sensor and motor delays. In fact, it has been shown that for certain classes of problems models are strictly necessary (Conant and Ross Ashby, 1970; Francis and Wonham, 1976). A similar idea is employed in recursive Bayesian estimation like Kalman (Kalman, 1960) or particle filters (Liu and Chen, 1998) where each update step integrates predictions based on a model and prior knowledge of the state with measurements to generate output estimates of the state.

In contrast to cognitive robotics however, the internal models used in the context of control theory are typically mathematical systems of the system (the plant) and stated explicitly

when designing the controller for example as systems of differential equations. In robotics the goal is to have the robot learn the internal models — both the inverse and forward models — autonomously. An example of such a strategy would be self-exploration mediated through babbling (Der and Martius, 2006; Dearden, 2008; Schillaci and Hafner, 2011; Baranes and Oudeyer, 2013; Martius et al., 2014).

Tin and Poon (2005) compare the concepts of internal models from disciplines of biology and control theory (adaptive control) and show how these concepts correspond. Interestingly, almost all concepts known from the biological context can be shown to have an engineering equivalent in control theory. This conversely means that metrics known from control theory like stability, robustness or convergence could possibly be applied to the analysis of biologically inspired internal models.

3.2.3. Robotics

A lot of the ideas revolving around internal models from biology as well as control theory have found their way into the field of robotics. Thus, only a small selection of work, which is interesting in the context of this work, will be discussed here.

It has been suggested that the idea of coupled inverse and forward models can be used for motor control as well as action recognition (Wolpert et al., 2003). This idea has successfully been implemented in robotic systems (Demiris and Khadhouri, 2006; Schillaci et al., 2012b, 2013) employing the internal models of its own robot body to recognize the actions of other individuals by means of the prediction error of these models. Demiris (2007) shows the connection of this approach to psychology discussing descriptive (recognition from data) and generative (internal model) approaches.

Most internal (forward) models consider only short predictions into the future in a tight sensorimotor loop. Internal simulation goes further than that in the sense of being more complete — often also incorporating the environment and other agents — and by simulating further into the future. In principle, short predictive steps can be chained so there is no clear border between the two extremes. An investigation into multi-step predictions can for example be found in Ziemke et al. (2005). A similar experiment was conducted by Hoffmann and Möller (2004); Hoffmann (2007) concatenating the same forward model by feeding its sensory output back into the input. In this work, internal simulation is used only as a name for a more complicated internal prediction.

An internal simulation can, together with the corresponding inverse models and facilities to generate and try out actions, provide a robot with a „functional imagination “ (Marques and Holland, 2009). This is to some extent related to the simulation theory of cognition (Hesslow, 2012), which is more general and rejects the idea of explicit (symbolic) internal models. Holland and Goodman (2003) outline how and which kind of internal models are compatible with the simulation theory of cognition[1]. Barsalou (2009) presents further evidence from

[1]The internal models in the sense of the ones defined by Wolpert et al. (2003) turn out to be of the class of compatible ones. Holland and Goodman (2003, p. 4) write:

> For example, in the field of artificial neural networks, a network that has been trained to produce a particular set of outputs in response to a given set of inputs is often said to have learned „an internal model “ of the problem; this internal model is simply a pattern of synaptic weights that

psychology and cognitive science for the hypothesis that simulation plays a central role in cognition.

The idea of using an actual simulator instead of learned models have also started to be explored in the field of robotics since the computational power of robots has been increasing steadily. For example Bongard et al. (2006) show a four-legged robot that can use an explicit internal simulation for the tasks of self-modeling and action generation for locomotion. Using this architecture, the robot can recover from physical damage by re-learning its new self-model, generating a new gait for the changed morphology. Similarly, a swarm of robots has been shown to be able to use internal simulation of the robot and its environment to evolve robot controllers, which are then used on the real robots (O'Dowd et al., 2011). This system has been shown to be able to adapt to changes in the environment.

3.3. Internal Model Representation: Machine Learning Methods

In the context of (cognitive) robotics, internal models are, in contrast to control theory where models are most of the specified analytically for example in the form of differential equations, often learned from experience, i.e., collected sensorimotor data. These models are then represented using machine learning methods.

An example for a learning task would be learning the forward model of one arm in the sensory space of a robot observing the arm with a camera. This forward model would then map motor values, e.g., joint angles to the endeffector position in the space of its camera, i.e., xy-values in the pixel grid of the camera. This model can be learned from previous exploration of the sensorimotor space, i.e., pairs of (joint angles, xy-pixel values), generated by for example motor babbling (Demiris and Dearden, 2005; Schillaci and Hafner, 2011; Baranes and Oudeyer, 2013).

For a broad overview of the current state of the art, Nguyen-Tuong and Peters (2011) review machine learning models for sensorimotor tasks and focus (mainly) on learning forward models. Different types of models and how they can be learned as well as what the challenges in a robotics context are, are discussed. Three case study applications are presented in detail.

In the following, the models used in this thesis will be discussed shortly. For more in-depth treatment of the specific methods refer to the references. All discussed models are of the class of supervised regression models.

Besides the ones discussed here, there are countless further general machine learning methods. In the field of learning of internal models, there are other prominent classes of models not used in this thesis of which general probabilistic models like bayesian networks, self-organizing maps, convolutional neural networks, recursive neural networks and other

happens to give the correct outputs, and it is only in rare cases or in certain special types of networks that the characteristics of the internal model can be related explicitly to the characteristics of the problem.

The criticism of Hesslow (2012) is more directed towards internal models in the Good Old-Fashioned Artificial Intelligence sense which are mostly symbolic with explicit semantics (also see the discussion on grounding section 3.1) and cater towards logical manipulations.

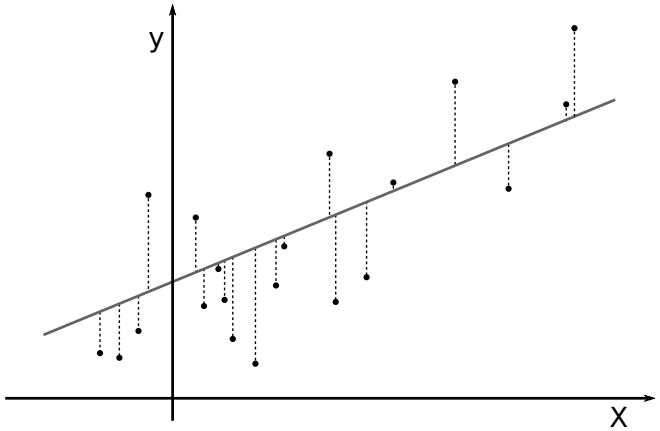


Figure 3.2.: Ordinary linear regression with data as points, grey model fit and dashed redidduals

connectionist methods. More recently, deep learning methods have started to gain importance as these methods start to surpass human skill levels in areas previously thought to be difficult for algorithms (Mnih et al., 2013; Schmidhuber, 2015; He et al., 2015; Mnih et al., 2015). A good introduction for the use of probabilistic models, specifically Bayesian networks, is given by Dearden and Demiris (2005).

Linear regression via for example ordinary least squares goes back to Gauss (1823) and can be readily extended to the multivariate case (Mardia et al., 1979). A simple graphic illustration of the basic process is depicted in fig. 3.2. Often these models are regularized using Tikhonov regularization (Tikhonov, 1943) and are then also referred to as Ridge Regression (Hoerl and Kennard, 1970). This practice is especially useful if the problem is ill-posed.

In contrast to the parametric model of linear regression, a Gaussian process is a nonparametric model. Using a Gaussian process for a regression task means to perform the regression

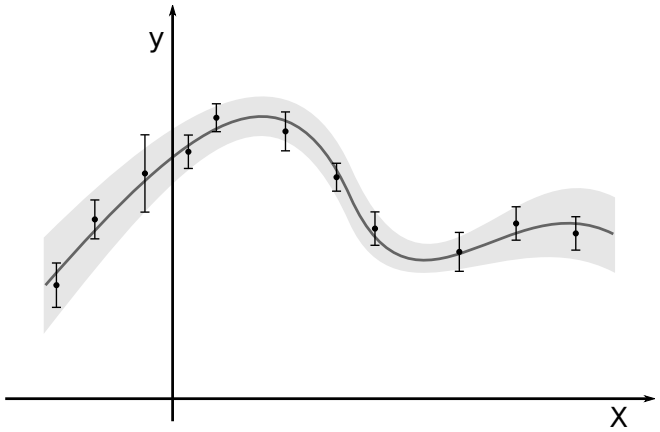


Figure 3.3.: Gaussian process with data as points with error bars, grey best model and light grey 95% confidence bands

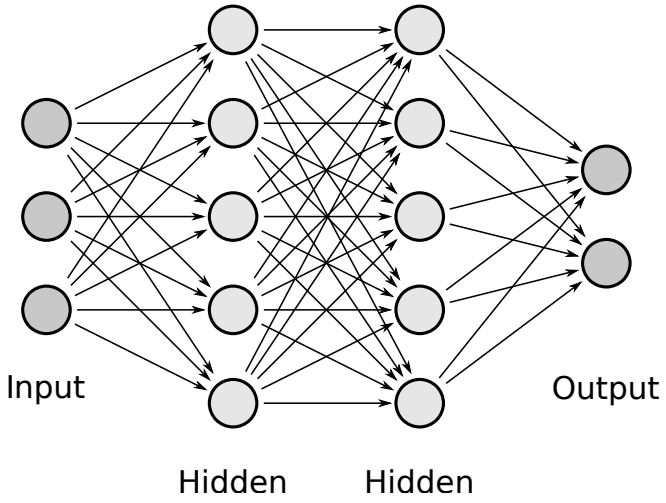


Figure 3.4.: Multilayer Perceptron (MLP)

in the space of all possible functions as opposed to performing the regression with a restricted class of functions as done for example by using linear regression (Rasmussen and Williams, 2006). Given a prior probability for all possible functions and the data points this methodology yields a posterior distribution over all possible functions. This method is illustrated graphically in fig. 3.3.

The k-Nearest Neighbors (k-NN) algorithm can be used for classification and regression. This algorithm operated directly on the training data insofar as the output consists of either the majority vote or the average of the k nearest (different metrics are used in practice) neighbors to the point to be classified or predicted respectively (Cover and Hart, 1967). A commonly used variant is to use all neighbors inside a fixed radius (Bentley, 1975) and possibly to use the distances as weights for the different neighbors.

Multilayer Perceptrons (MLPs) are feedforward artificial neural networks consisting of several layers of neurons where each neuron is a nonlinear activation function, usually a sigmoid. The layers are usually fully connected via variable weights which can be trained by for example backpropagation (Werbos, 1974; LeCun, 1985; Rumelhart et al., 1985) or RPROP- (Riedmiller, 1994). A schematic example of a MLP is depicted in fig. 3.4.

Support Vector Machines (SVMs) were introduced for classifications. They construct the best hyperplane(s) separating the points contained in the classes to be classified. A two-dimensional example of this process is depicted in fig. 3.5. Since most problems are not linearly separable, the input space is usually mapped to a higher, possibly infinite dimensional, space using the kernel trick (Vapnik, 1963; Vapnik and Chervonenkis, 1971; Cortes and Vapnik, 1995). SVMs can also be extended for function estimation. Schölkopf and Smola (2002) give a more recent overview on the topic and Smola and Schölkopf (2004) contains a tutorial and discusses some practical aspects of using SVMs for regression.

If not stated otherwise, the following libraries were used in this thesis: for the implementation of most machine learning algorithms as well as pre- and post-processing scikit-learn

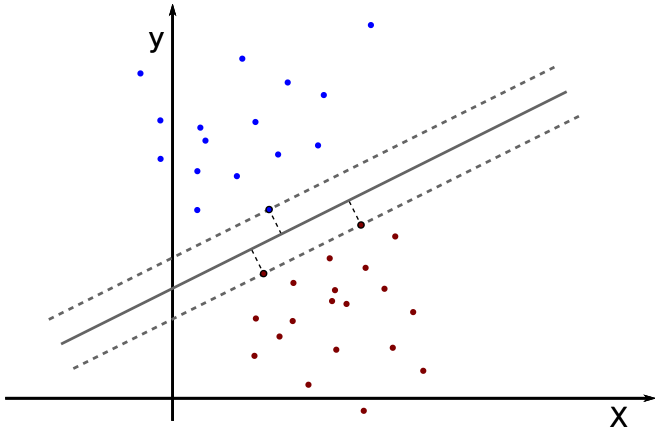


Figure 3.5.: Support Vector Machine (SVM)

(Pedregosa et al., 2011) was used. MLPs were implemented using PyBrain (Schaul et al., 2010) and linear regression was done using SciPy (Jones et al., 01). Furthermore, Hyperopt (Bergstra et al., 2013) was used for hyper-parameter optimization, pandas (McKinney, 2011) for working with time series data, and Matplotlib (Hunter, 2007) for visualization.

Part II

Network Robotics with Single Robots

Chapter 4

Introduction and Related Work

4.1. Introduction

Wireless networks, ranging from the ubiquitous cellular networks to ad hoc networks used for example in car-to-car communication, are already a crucial part of our communication infrastructure. Furthermore, the paradigm of the Internet of Things will add network capabilities to many objects, often via wireless interfaces (Mattern and Floerkemeier, 2010), which will then commence in machine-to-machine communication. These wireless network connections can in turn span entire (ad hoc) wireless networks between networked smart objects. One example of this technology already in deployment today are Wireless Sensor Networks (WSNs).

In addition to these machine-to-machine communication scenarios, very flexible, easily deployable wireless communication and/or sensor networks, which can be used for example as backup communication infrastructure, can be composed of robots as mobile nodes. These networks are needed for example in disaster scenarios as a robust ad hoc communication infrastructure when local communication infrastructure is (partly) unavailable. This approach has been explored in projects like SMAVNET (Hauert et al., 2010b) or AirShield (Daniel et al., 2010).

Communication capabilities between robots and between robots and humans, for example via hand-held devices, are also needed for semi-autonomous operation of robots for instance in disaster scenarios in a supporting role for firefighters or rescue crews.

In WSNs specifically, mobile robots can function as mobile nodes, i.e., as part of the network, or as users of the WSN to harness the WSN to extend their own sensing capabilities. As mobile nodes, robots can be used as a means of deployment, to dynamically adapt the network to new tasks and/or situations, to optimize the WSN for different metrics and to repair the WSN, especially for spatially correlated failures, i.e., several nodes are failing in one place for example after an earthquake or fire.

This area is sometimes called network robotics. In this work, the term network robotics will be loosely defined as the area of research concerned with mobile robots navigating in, integrating into, or forming, a wireless network.

Integrating mobile nodes into wireless networks poses interesting challenges specific to wireless networks in the technical design of network protocols etc., such as the characteristics of delay tolerant networks or constantly changing network topologies in networks with

mobile nodes. This work however focuses on the physical aspects of integration only, i.e., with the physical layer (PHY) of the OSI model, because it poses the most fundamental problems of network robotics as a working PHY connection is needed by all higher layers of the OSI model to function properly. This includes tasks such as establishing and maintaining a stable network connection by ensuring good signal quality, which for a mobile robot means moving to a place with good signal reception. As will be shown later, methods developed to solve those basic task can often be adapted to meet goals from other layers such as maximizing packet throughput rates possibly in a cross-layer approach.

Thus, integration into these networks also means first of all physical integration of the robot. Wireless communication offers more degrees of freedom for integration than wired communication as a mobile robot can freely move and position itself in space. However, the need for good reception, i.e., a good physical signal-to-noise ratio (SNR), still poses some restrictions on the positioning of the robot depending on the requirements of the particular task for instance a minimal required packet throughput. Because of effects such as shadowing and multi-path fading (see section 2.3), different spatial locations have different SNR even inside the communication range of a node. Robots integrating into these wireless networks have to be able to cope with these modalities.

This fact is even more pronounced since all wireless technologies share the air as a physical medium. Many of these technologies work in unlicensed ISM bands (see section 2.2.2) which implies a lot of competition for bandwidth between network technologies and also between users. This in turn leads to very noisy and non-stationary characteristics of the network parameters of these networks, which is can be challenging for new algorithms but also presents interesting dynamics which can be explored and exploited.

For a more complete and in-depth discussion on the subject of wireless networks and their dynamics refer to chapter 2.

For static nodes many algorithms and techniques exist that try to deal with these challenges. However, robots as mobile network nodes offer new ways of dealing with these challenges by exploiting sensorimotor interaction in the sense of using for example measured SNR as a sensory modality. Most of the time this means the entire robot to physically moving — in contrast to other sensorimotor tasks like for example grasping for which the robot mainly actuates the arm and hand joints — and makes use of correlations between the gathered sensory data and knowledge about its motion.

This means that a robot can use its mobility to mitigate negative effects to deal with the issues of signal attenuation, noise and interference, which are all of a spatial nature, in a cross layer like fashion in the sense of the physical actions of the robot beginning below the PHY level. This can in principle encompass everything from bandwidth optimization over energy savings up to interference mitigation, for example moving away from a stationary node from a different wireless network which acts as source of interference.

These tasks should be solved in an autonomous manner and thus no prior knowledge about the network, i.e., topology, physical location of nodes, etc., should be necessary for the algorithms. Additionally, all algorithms should only be based on local information in order to enable scaling to multiple robots and arbitrary network sizes, i.e., possibly networked swarms.

Besides being a technical challenge because of their dynamic nature, wireless networks can also be used as an interesting testbed for taxis algorithms. The measured SNR of a wireless connection is a scalar field and is thus very similar to for example chemical concentrations as measured during chemotaxis. Using real chemical concentrations in experimental setups is not very practical because of their volatile nature and susceptibility towards air movements (as for example induced by the experimentator). Therefore, it is desirable to use other scalar fields instead. Often light fields, i.e., measured intensity of a light source at different positions, is used instead. However, light intensities are very stable and smooth and do not show interesting dynamics. As discussed above, measured SNR of a wireless connection does show challenging dynamics. Basically, these dynamics consist of a deterministic part varying at the scale of the wavelength of the signal and of a stochastic part induced by external interference and sources like moving persons etc. (for details, see section 2.3). The typical length scale for variation in these measurements is of the order of the wavelength, i.e., of the order of $12.5\,\mathrm{cm}$ for $2.4\,\mathrm{GHz}$, which is also a very convenient length scale for typical robot sizes. Thus, studying the integration of robots into wireless networks is not only an important technical task with practical relevance but also an interesting scientific task relating to all kinds of algorithms dealing with scalar spatial quantities such as taxis algorithms.

4.2. General Related Work

As discussed, the focus of network robotics is mainly on algorithms and problems dealing with physical aspects of wireless networks. This mainly means localization of nodes, which is the most studied subject in this area. Nevertheless, some other related approaches and problems such as source seeking or communication-based swarming are also discussed in this section. Algorithms for self-organized deployment, coverage optimization, repair etc. are investigated in section 9.3. Furthermore an overview of recent testbeds using mobile nodes, (ground-based and/or flying) often in the context of WSNs, is given in section 9.4.

Localization of network nodes is a topic which has been extensively studied in the past for wireless (sensor) networks (Patwari et al., 2005). Most of the related research is concerned with static networks and cooperative localization of the static nodes or with localization of one mobile node using the knowledge of the static nodes. A very comprehensive survey was presented by Wang et al. (2010) and deals with all kinds of possible algorithms with and with out anchors and/or mobile nodes as well as all kinds of sensing modalities. As stated in the previous sections this work, however, focuses on the inverse problem of one mobile node for example attempting to localize other (possibly static) nodes using no prior knowledge. Nevertheless these algorithms are interesting since they deal with similar technical problems.

4.3. Localization with Mobile Nodes

The task of localizing a network node in a network consisting only of static nodes has been studied extensively. Mao et al. (2007) presented a comprehensive review on the topic. Besides algorithms for localizing static nodes, there exist some algorithms making explicit use of the mobility of the nodes. Even though these algorithms often originated in the networking

community, they are converging more and more towards techniques already known from robotics as there are increasingly more roboticists becoming versed in the topic.

An interesting example of minimal actuation is the work presented by Elnahrawy et al. (2007) who added rotatable directional antennas to fixed nodes to combine angular and RSSI measurements to localize nodes. Even though the nodes are not mobile, using rotatable directional antennas can be seen as a very simple kind of sensorimotor exploitation. Furthermore, rotatable directional antennas could be used on mobile nodes to improve the sensing capabilities by adding angle measurements.

A rather sophisticated way to directly measure bearings without antennas specifically designed to be directional was presented by Derenick et al. (2011). They exploit the fact that real antennas are, in contrast to idealized theoretical ones (see section 2.2.3), not perfectly isotropic. They formulate an algorithm to determine bearings by rotating these imperfect antennas and use the resulting ambiguous measurements to localize a team of mobile robots by repeated measurements while moving.

If no direct angular, i.e., directional, measurements are available, one method to overcome these limitations is to substitute directional measurements by calculating gradients from spatially referenced measurements using for example additional GPS measurements. Han et al. (2009) show how such gradients can be used to localize network nodes by fitting a locally linear model to the measured signal strength data. Combining several such directional measurements at different points then leads to an estimated localization. Estimating gradients by fitting a locally linear model to data is also employed in an algorithm presented by Paul et al. (2011). They explicitly show how this procedure of fitting a locally linear model effectively mitigates the effect of noise.

In a similar approach, Dantu et al. (2009) estimate spatial gradients in RSSI measurements to gain bearing information. In contrast to Han et al. (2009) however, they only rely on local odometry instead of global GPS measurements. Furthermore, they use a finite difference like sampling pattern and calculate a principal component analysis (PCA) instead of fitting a locally linear model to the measurements.

More complex models than locally linear approximations have been shown by Fink and Kumar (2010) to effectively estimate source localization. This algorithm uses Gaussian process models based on path-loss and attenuation priors and combines them with RSSI measurements, odometry and information from a laser range finder. The algorithm performs very well in terms of accuracy and statistical efficiency, i.e., the number of samples needed but it scales cubically with the number of samples.

4.4. Source Seeking Algorithms

A more straight forward task than source localization is source seeking, which means moving towards the location of the wireless signal source. This class of algorithms is related to general taxis algorithms like chemotaxis. The different taxis algorithms in the literature mainly differ in what kind of sensory input, especially odometry, is available.

In chemotaxis (Adler, 1966; Berg and Brown, 1972; Lux and Shi, 2004) where bacteria are moving to the point of highest concentration of food molecules (or to the point of lowest

concentration of harmful molecules), a very similar problem to source seeking is solved. However, because of their size, bacteria cannot directly measure gradients in the concentration of the molecules and do not have access to odometry information. Nevertheless, they evolved behavior suited for coping with these restrictions. While these algorithms are effective without using gradient information, they are less efficient because of their stochastic nature.

When directional, i.e., bearing information is available, simple Braitenberg-style algorithms can lead to source seeking behavior. In the most simple instantiation, this is achieved by connecting two directional antennas via a very simple artificial neural network directly to the two motors of the differential drive of a robot (Braitenberg, 1986).

A more complex algorithm for source seeking is presented by Wadhwa et al. (2011). They show a multi-phase heuristic algorithm used to mitigate the effects of very flat gradients far away from the source. The algorithm starts by estimating a rough orientation in relation to the source and then estimates the correct direction towards the source. Then the robot moves in that direction while measuring RSSI values as long as a moving average of these measurements is increasing. This algorithm can be seen as a heuristically improved version of chemotaxis.

A frontier based algorithm for source seeking is presented by Twigg et al. (2012). In contrast to Wadhwa et al. (2011), RSSI gradients are explicitly estimated from RSSI measurements by fitting locally linear models to the measurements. They also show an interesting way to update these estimates continuously as new measurements are generated. They combine these gradient estimates with a moving average of the RSSI and a frontier based approach for navigating. This means that no prior map of the environment is needed but one is generated while executing the algorithm and moving towards the source.

A taxis algorithm for general abstract taxis with the exemplary use for source seeking of a wireless signal source is presented by Atanasov et al. (2012). This algorithm is based on Random Direction Stochastic Approximation (RDSA) known from the stochastic approximation literature (Kushner and Yin, 2003; Spall, 2005). The noise characteristics of the physical model are however discussed only very briefly and they fail to mention motor noise at all. As discussed in chapter 7, small scale fading is especially problematic and can violate some of the convergence conditions if not dealt with correctly (see specifically section 7.3.3).

In general, gradient based methods can be proven to converge for the case of signal strength measurements in wireless networks under some constraints (Atanasov et al., 2012; Blum and Hafner, 2014). These methods basically implement a gradient descent algorithm directly on noisy measurements and need to mitigate local maxima created by small scale fading.

4.5. Communication Based Swarming

(Nembrini et al., 2002) present a minimalistic algorithm for coherent swarm taxis. For the robots, a unit disk connectivity model, i.e., omnidirectional connectivity, with disk radius much smaller than swarm radius, is assumed. The robots also follow an avoidance behavior using infrared sensors. A call-answer mechanism is used to check for connectivity, and a loss of connectivity leads to a 180° turn of the robot. This algorithm could be shown to lead

to emergent swarm coherence also in the presence of obstacles and when moving towards a beacon.

A similar algorithm was shown for flying robots in the Airshield project (Daniel et al., 2010). In this work the robots, however, have access to GPS information so they can easily turn around correctly once they disconnect from the cluster. The resulting movement direction is formulated as a virtual force. Furthermore, a cluster fusion and an overlay movement virtual force are introduced. Together these virtual forces lead to a coherent steering strategy for the swarm.

In the context of the project SMAVNET (Hauert et al., 2010b), using simple and lightweight and thus safe fixed wing flying robots, several algorithms using RSSI measurements as sensory inputs were developed. A genetic algorithm was for example employed to evolve a swarming algorithm using only topological and no position algorithm (Hauert et al., 2009). An algorithm based on logarithmic spirals to reconnect a disconnected robot (Hauert et al., 2010a) was also developed in this project.

There are many examples for swarm algorithms based on communication from the WSN context for example node deployment, coverage optimization or network repair. These algorithms are reviewed separately in section 9.3.

Chapter 5

Real World Measurements of Network Parameters

5.1. Introduction

To gain further insight into the real world dynamics of wireless networks, measurements were conducted in real environments. One such measurement was already discussed in section 2.2.2 dealing with spectral usage of ISM bands. These measurements showed complete spectral energy signatures measured for a single point over a multi-day time frame. This included all kinds of wireless technologies from remote controlled vehicles over wireless keyboards to IEEE 802.11 traffic. In this chapter, the characteristics of IEEE 802.11 wireless networks are the focus. Other wireless technologies mainly show up as external interference.

5.1.1. Measurement Modalities

Most of the time, physical parameters such as RSSI are the most interesting ones from a robotics point of view because of their purely physical nature. As such they can be used for navigation purposes since they are directly dependent on the spatial location where they are measured. Sometimes, however, other more high-level network parameters such as packet delivery rate (PDR) or packet error rate (PER) are of interest if the goal of an algorithm is for example to optimize the network for throughput between two specific nodes. These parameters in general have nonlinear inter-dependencies, especially between RSSI and for example PDR where a small change in RSSI can lead to a very big change in PDR because of how the protocols on higher OSI levels are implemented.

5.1.2. Measurements

This chapter mainly presents two types of measurements: outdoor large scale measurements using a hand-held laptop with external antenna and indoor small scale measurements using robots. For practical reasons, outdoor measurements were not conducted using robots even though it would lead to more precise spatial measurements. Large scale measurements mainly show the overall characteristics of the parameters while indoor measurements are mainly of interest to understand small-scale effects such as multi-path small-scale fading and attenuation due to obstacles such as walls.

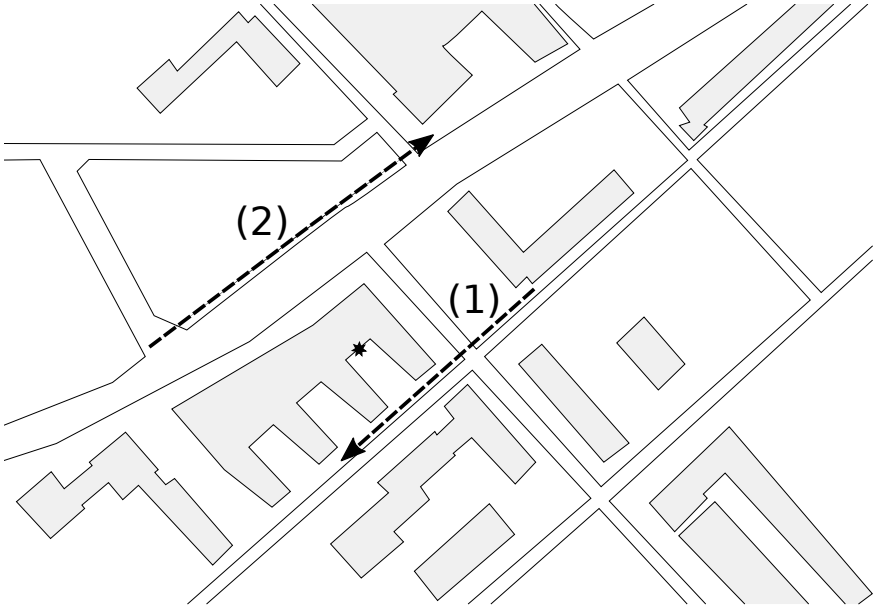


Figure 5.1.: Measurement paths 1 and 2 at the Humboldt-Universität Adlershof campus. Buildings are gray and the fixed HWL node is represented as a star.

5.2. Outdoor Large Scale Measurements

Outdoor measurements were conducted in the context of the Humboldt Wireless LAB (HWL) (Zubow and Sombrutzki, 2011) which is a large-scale wireless mesh network installed on the Humboldt-Universität zu Berlin Adlershof campus. It consists of around 100 indoor and outdoor nodes. The outdoor network covers almost the whole campus enabling extensive outdoor experiments. Only outdoor nodes were used in the experiments.

5.2.1. Hardware Setup

A laptop was equipped with a USB wireless adapter (Atheros AR9271) and a GPS receiver to track its movement. It was sending 800 User Datagram Protocol (UDP) broadcast packages per second with its current GPS position. A stationary HWL node (Atheros AR5414) received and analyzed the packets saving the GPS position, time-stamp, RSSI and network parameters. This data was then analyzed and visualized after the experiment. This laptop was carried along the planned trajectory manually.

5.2.2. Experimental Setup

The position of the HWL node and the university buildings as well as the paths for the two experiments are depicted in fig. 5.1. Note that even though on this sketch the whole building seems to be blocking the line of sight between the HWL node and measurement path 2, there are actually only two panes of glass (which should have negligible attenuation) in the line-of-sight path of the signal. This can be seen in detail together with the actual position of the

Figure 5.2.: Placement detail of the stationary HWL node

HWL node in fig. 5.2, which depicts a photograph of the situation.

5.2.3. Results

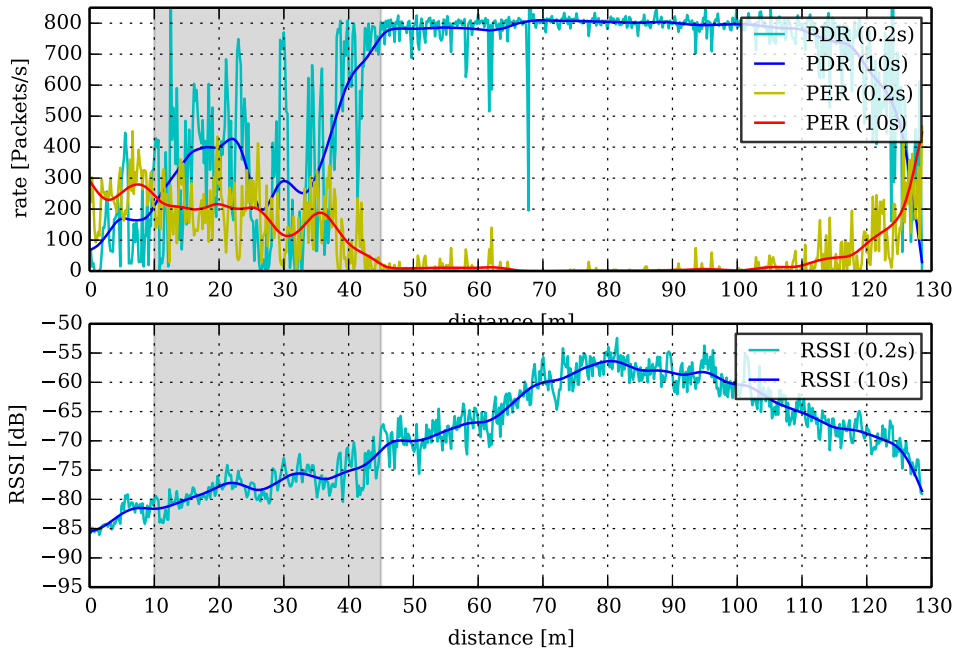
Measurements for RSSI, PDR and PER, which counts packets with a checksum error, for two measurement paths as shown in fig. 5.1, are depicted in fig. 5.3a and fig. 5.3b respectively. The graphs shows measurements calculated for windows of $0.2\,\mathrm{s}$ and smoothed with a Hanning window function with a width of $10\,\mathrm{s}$.

Both measurements show a very noisy behavior even though the rates were calculated with rather large $0.2\,\mathrm{s}$ wide bins. This is to be expected for outdoor measurements because of the high potential for external interference in a university environment. Furthermore, the physical environment is highly non-stationary since persons and vehicles are moving around.

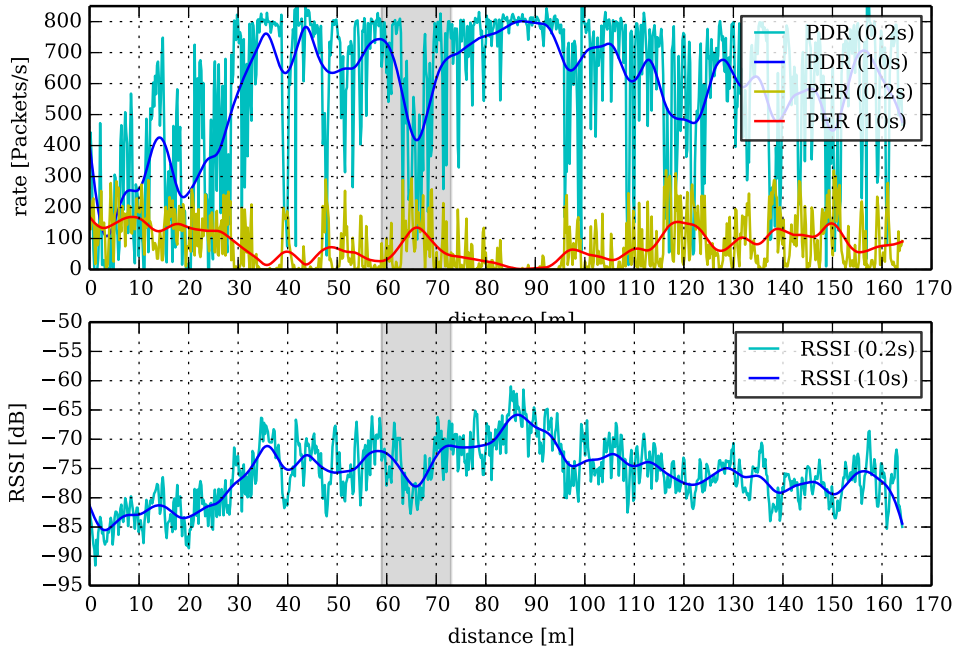
The most interesting part of the first measurement is marked as a gray area. It shows how nonlinear wireless networks can be in real world scenarios. In this area, the RSSI rises steadily while the resulting PDR varies greatly and in a rather nondeterministic way. We suppose this was caused by external interference, but it is very hard to determine the actual cause since complete knowledge of the current state of all radio sources is never available.

The second measurement shows even more drastically how dynamic and noisy a real wireless environment is. The response of the PDR to the RSSI is almost everywhere nonlinear and there is poor correlation. To give an example, one of the drops of PDR, which is marked gray, was caused by a bus passing by.

In order to show that most of the dynamics and noise in the system is due to external



(a) Measurement path 1



(b) Measurement path 2

Figure 5.3.: Measurement path 1 and 2(see fig. 5.3a): PDR, PER, and RSSI are shown respectively. The depicted values are calculated for windows of 0.2 s and smoothed with a Hanning window function with a width of 10 s.

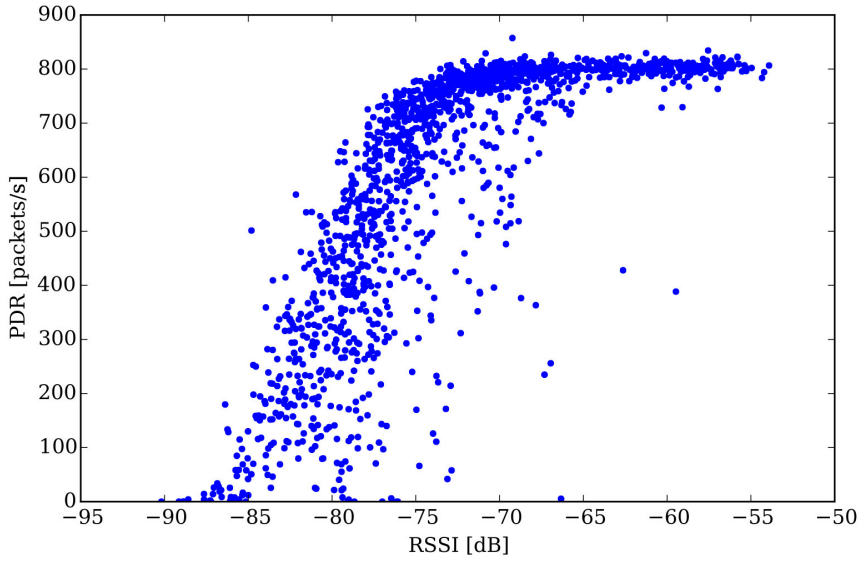


Figure 5.4.: Correlation of RSSI and PDR for all measurements combined

interference, the correlation of the PDR and RSSI is depicted in fig. 5.4. The plot shows that for most RSSI values, there is a big variation in PDR ranging from almost no received packets to an almost flawless PDR. This is a good indicator for external interference.

5.2.4. Conclusion

The dynamic and noisy nature of wireless environments has implications on the design of algorithms for network robotics. Using only knowledge about some network parameters such as RSSI only is insufficient. The problem has to be tackled as a whole using multi-dimensional data, especially the combination of spatial locations and network parameters. The most promising way to deal with these problems is to make use of the correlations in the data generated by sensorimotor interaction (Pfeifer et al., 2007). Actively shaping the sensory information by means of moving the robot in the wireless environment can reduce complexity and the problem can become manageable.

5.3. Indoor Measurements: Small Scale Fading

For mobile robots navigating in wireless networks, knowledge of physical characteristics on a small scale are crucial. Ground-based robots in particular often move distances of the order of centimeters so effects on this length scale are of importance. As discussed in section 2.3.2, the main effect on these length scales is multi-path small scale fading, i.e., interference of the same signal with itself due to different travel times while traveling different physical paths to the same location. This effect is particularly pronounced in indoor environments because of the abundance of objects on which the wireless signal can reflect and scatter and thus create many different physical paths to the receiver.

Figure 5.5.: The Robosoft RobuLAB 10 robot base used with attached plastic pipe to elevate the USB wireless adapter above the ground. In this photo only the robot base without the laptop computer and USB wireless dongle is shown.

5.3.1. Experimental Setup

A Robosoft RobuLAB 10 as depicted in fig. 5.5 was used as a mobile base because of its highly precise odometry. A standard laptop fixed on top of the robot was used for data logging using a USB wireless adapter mounted on a plastic pipe well above the ground for measurements. A second stationary laptop, also with a USB wireless adapter placed well above the ground, was used to generate packets to be received and measured on the robot.

All experiments were conducted in an office environment (a map is depicted in fig. 5.6). The stationary node was placed in one of the offices and the robot moved along various randomly chosen straight paths with a constant speed of $2\,\mathrm{cm\,s^{-1}}$. Simultaneously, the stationary laptop was sending packets to be received and measured by the robot.

Two types of experiments were conducted: one over a longer distance (along the long corridor of the office) and several shorter ones in varying directions at different, randomly chosen places in the office. Since the actual configuration of the offices (furniture, people, wall types, etc.) are unknown, the specific location is not relevant here.

5.3.2. Results

The first measurement was conducted along a $10\,\mathrm{m}$ long straight line to capture all the effects of the wireless medium. RSSI was measured and the values reported by the driver of the wireless adapter are plotted directly in fig. 5.7 as cyan dots. The values were extracted from

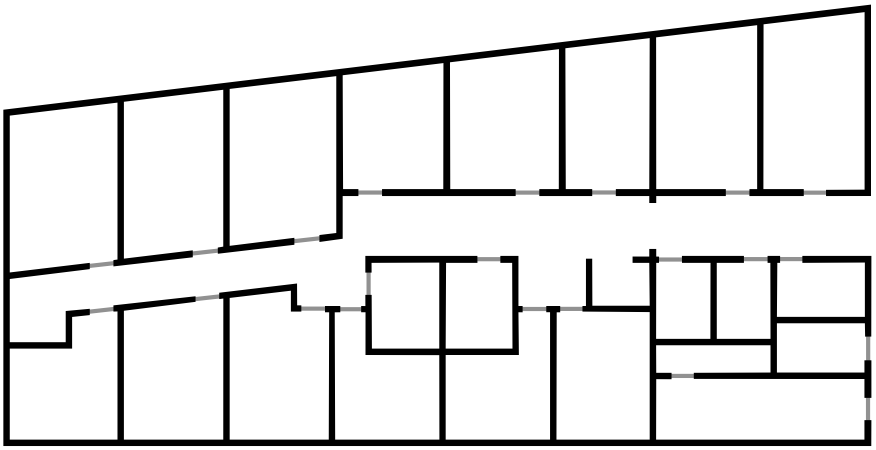


Figure 5.6.: Floor-plan of the office environment for the indoor experiments

the Radiotap[1] generated by tcpdump/libpcap[2]. As can be seen, the measurement units are unfortunately only integer dB. Thus, also moving averages are shown.

As discussed in section 2.3 and as already seen in section 5.2 the effect on the largest spatial scale is path loss, which is the general trend seen in fig. 5.7 from lower to higher RSSI values. The moving average with a window of $1\,\mathrm{m}$ also shows this trend as a black line.

The effect of multi-path small scale fading can also be seen in this measurement using a moving average over $1\,\mathrm{cm}$, which is plotted as a blue line. It shows variations on the scale of the wavelength of the $2.4\,\mathrm{GHz}$ ISM band used by IEEE 802.11g, which is $12.5\,\mathrm{cm}$ (these small scale effects can also be seen more clearly in fig. 5.8).

The raw measurements indicated as cyan dots show additional noise of the order of $5\,\mathrm{dB}$.

[1] `http://www.radiotap.org/`
[2] `http://www.tcpdump.org/`

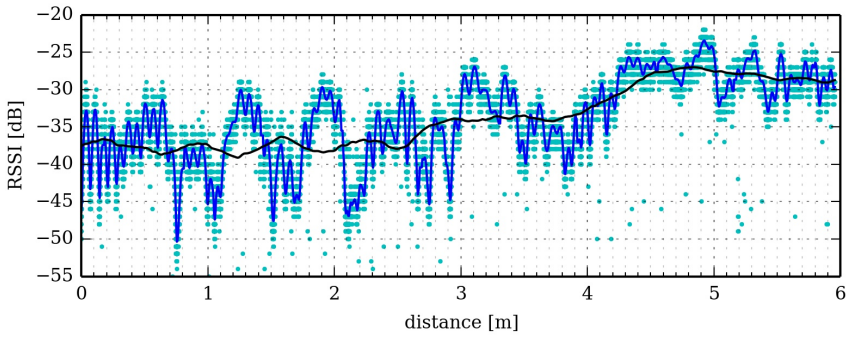


Figure 5.7.: Large scale indoor RSSI measurements on a straight line with the RobuLAB 10 in an office environment. Raw measurements are shown as as cyan dots, a moving average with window size of $1\,\mathrm{cm}$ as a blue line and a moving average with window size of $1\,\mathrm{m}$ as a black line.

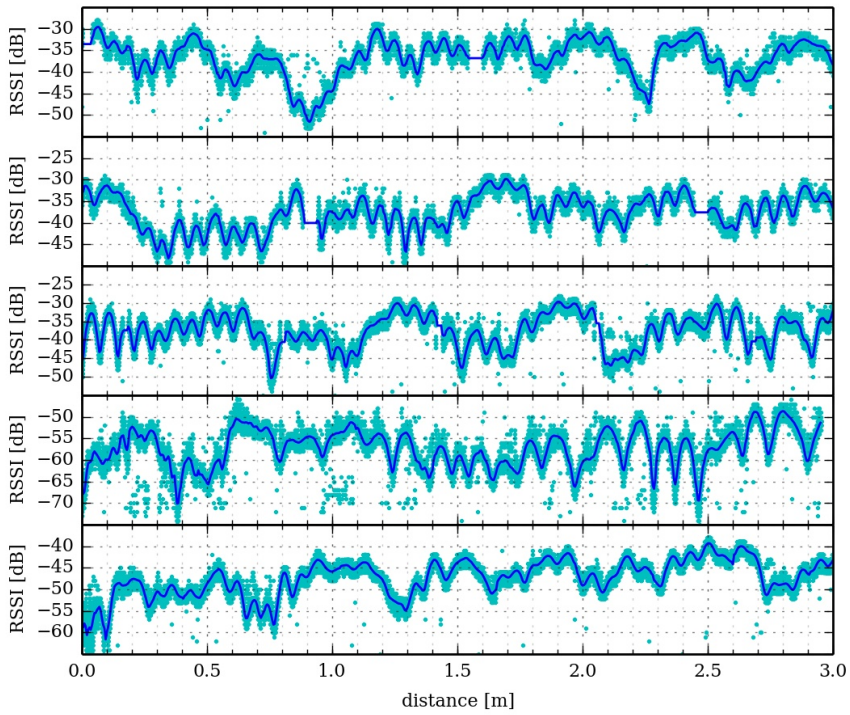


Figure 5.8.: Small scale indoor RSSI measurements on straight lines of random direction and location done with a mobile robot in an office environment. Raw measurements are shown as cyan dots and a moving average with window size of 1 cm as a blue line.

This noise is very time- and location-dependent because it is mainly due to external interference.

Since the small scale effects are especially interesting for ground-based mobile robots, another set of measurements on a smaller scale was conducted. The results are shown in fig. 5.8. These measurements were done on straight lines of 3 m length of random direction and location in the office environment to show as much variation as possible.

For these measurements, the large scale moving average was omitted. The small scale moving average again has a window size of 1 cm and clearly shows variations on the scale of the wavelength of the carrier wave. As expected for multi-path interference, they are deterministic and sinusoidal. Again, noise is of the order of 5 dB. Note that on these length scales, small scale fading clearly dominates over the large scale pathloss and is able to create local minima in the measured RSSI values. Algorithms dealing with spatial variations of RSSI thus have to be able to deal with these deterministic[3] variations.

[3]This is only true for stationary nodes and one mobile node. If the source node is also mobile this is no longer true.

Figure 5.9.: TurtleBot 2 with a USB wireless adapter

5.4. Indoor Measurements: Signal Strength Map

The experiments discussed up to now were purely one-dimensional for the sake of a better understanding of the details of the network parameter dynamics. When working on the experiment discussed in chapter 8, a lot of spatially referenced RSSI measurements were collected. Plotting all of these measurements on a two-dimensional map is informative for general trends in the RSSI map. Since those measurements were created over a period of several days while the office was in use (people moving around, doors opening and closing, etc.), only averaged values, i.e., deterministic large scale effects, are shown here.

5.4.1. Hardware Setup

The mobile robot TurtleBot 2, as depicted in fig. 5.9, was used as the mobile base for the measurements. It is outfitted with a regular laptop computer to which a USB wireless adapter (Atheros AR9271) was connected, which was placed approximately in the rotational center of the robot as can be seen in fig. 5.9. A second USB wireless adapter (Realtek RTL8192CU) connected to a stationary computer was used as the packet source. Both wireless adapters were elevated well above the ground to mitigate the risk of self-occlusion by the robot and occlusion by furniture close to the source.

5.4.2. Experiemental Setup

The floor plan of the office environment can be seen in fig. 5.10b. 17 individual experiments over a period of two weeks are summarized in the measurements. The position of the station-

ary node was fixed during this time interval but the office was in regular use, which means that the physical environment was changed on a local scale, i.e., moved furniture, opened and closed doors, and moving people.

5.4.3. Results

In total 422206 spatially referenced RSSI measurements were collected. They are depicted in fig. 5.10a. The bin-size for this histogram was chosen according to the precision of the spatial reference and larger than the scale of multi-path small scale fading.

As expected from the discussion about large scale pathloss in section 2.3.1, the RSSI decreased smoothly with distance from the source. The effects of the attenuation by the walls are not particularly pronounced, but that is likely due to the fact that the walls are not solid but simple dry walls. Noise levels were measured to be on average $4\,\mathrm{dB}$ with a logarithmic Gaussian distribution.

Note that the gradient of the RSSI field is very flat at the very ends of the long central corridor and most of the variation of the RSSI is close to the source. This fact is important for all gradient-based algorithms.

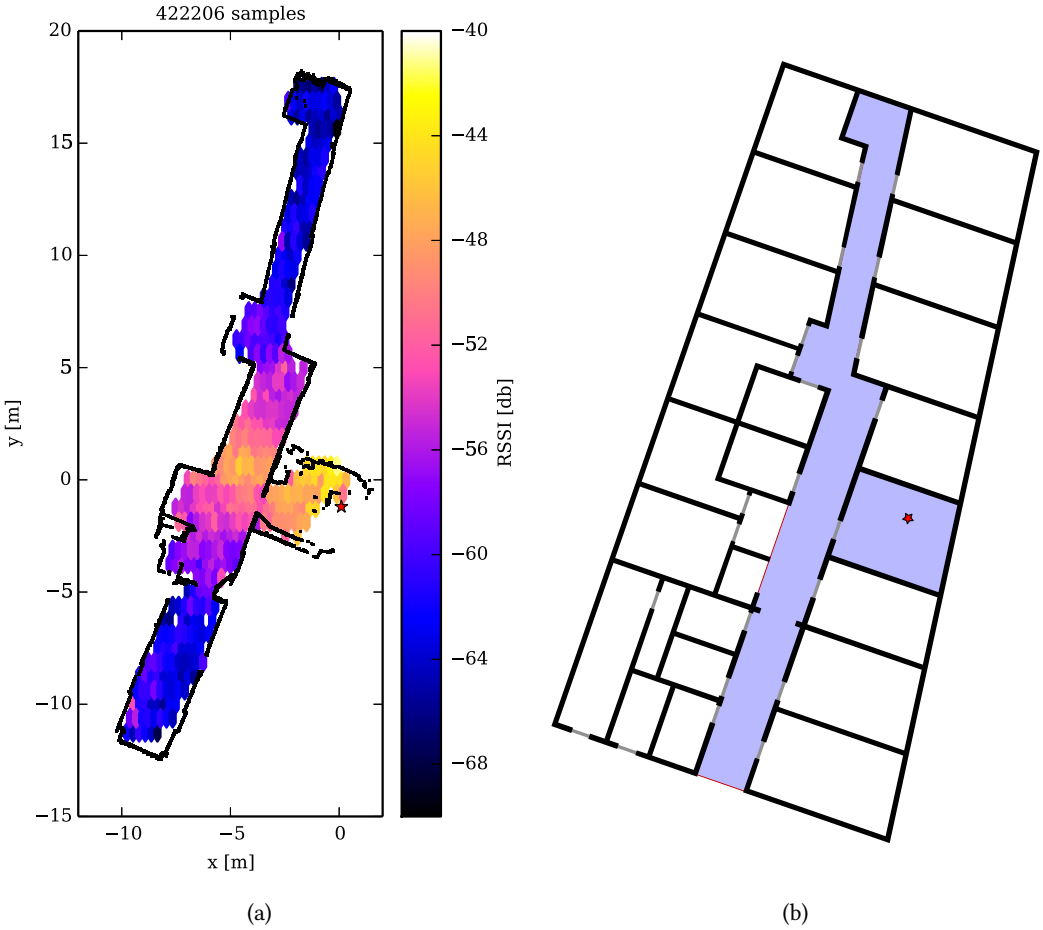


Figure 5.10.: Figure 5.10a shows the histogram of 422206 RSSI measurement samples for a single network node in an office environment collected with a mobile robot. The position of the static source node is depicted as a red star and walls are depicted as black lines. Figure 5.10b depicts the corresponding floor plan of the office environment in which the experiment was conducted. The area covered by the measurements is shaded in light blue and the position of the static source node is depicted as a red star.

This work was presented in (Blum and Hafner, 2013)[1].

Chapter 6

Robust Exploration Strategies for a Robot exploring a Wireless Network

6.1. Introduction

Integration of robots into wireless networks is important for a number of scenarios as discussed in section 4.1. If the network to be integrated into is unknown, one of the first tasks is network exploration for which the most basic case is finding the physical outline of the network. This is an important task when, for example, the coverage of a sensor network has to be evaluated or when a mesh network is partly destroyed after a disaster. A robust algorithm to explore the outlines of a wireless network is proposed in this section. This algorithm is computationally simple, uses directional measurements of the connection to network nodes and uses RSSI measurements only in a very basic way, i.e., not as a distance estimation. This algorithm is very robust against different kinds of noise for both sensory inputs. Furthermore, accurate odometry information is not necessary.

6.2. Problem Statement

Assuming no prior knowledge about the network, the first step of integrating a robot into a wireless network is to gain knowledge about the physical and topological characteristics of this network. The first tasks to be solved to explore the network include finding the nodes constituting the network and spatially outlining the network. Some very basic knowledge about the position of the network is assumed to be available insofar as the robot knows in which general direction the network is located or that the robot is located in communication range of a network node.

Actual localization — in most cases trilateration — of nodes using only data from network parameters is a difficult task. These difficulties are caused by the nonlinear relationship between signal strength and node distance, which is especially pronounced in non-line-of-sight scenarios. Furthermore, most of these techniques use known positions of so called anchor-nodes, in this case the robot would have to know its own position by for example accurate

[1]For a detailed breakdown of the contributions of the different authors refer to section 1.5.

GPS measurements, which are not always available or desired. Thus, explicit localization techniques for the network nodes will not be employed for this algorithm.

Therefore, the algorithm presented here consists of measuring only the relative direction of the nodes and to use signal strength measurements only in a very basic and hence robust way. The direction measurements can be performed in any technically feasible way, two of which are gradient estimation (Han et al., 2009) and usage of directional antennas. The usage of signal strength should be restricted in a way which avoids the difficulties of actual localization techniques for the algorithm to be robust[2].

The robustness of the algorithm in terms of sensory noise for both directional and RSSI measurements is then determined. For this, a number of simulation runs is performed and a parameter study in the amount of noise for both parameters is conducted.

6.3. General Problem Setting

6.3.1. Robot Model

This algorithm is designed for a flying robot in open space which does not have to be concerned with collision avoidance related to walls or other objects. Furthermore, special dynamic characteristics of some type of flying robot such as fixed-wing flying robots are not taken into account but for simplicity a holonomic platform for instance a multicopter is assumed. In principle this algorithm could also be used for a ground-based robot in a semi-open environment such as on an open field or in an urban scenario with an open street-grid, for which it would have to be modified slightly to take into account the reduced freedom of movement. It is not very well suited for indoor use, however, due to the usually complex topologies of such environments. Effects such as shadowing by buildings and so on are not as critical since the robot tries to follow RSSI isolines regardless of their actual shape.

No global positioning or local odometry is needed for this algorithm. Nevertheless, gradient estimation algorithms may be based on odometry. Furthermore, odometry or self-localization might be necessary in order to use the information gathered by this exploration algorithm, like for example when measuring network coverage. In general, the accuracy needed for these kinds of tasks is in general much lower than the accuracy needed for navigation algorithms.

Directional information needed for this algorithm is not assumed to come from a specific source. The noise characteristics are assumed to be Gaussian which is true for a wide range of (virtual) sensors. For this scenario a directional antenna can easily be exploited because a holonomic robot is assumed. Gradient based direction estimation needs basic odometry but has been shown to also work well (Dantu et al., 2009).

[2] In section 6.6.2 the robustness analysis of the algorithm will show that the algorithm is in fact very robust against measurement errors in the direction measurements. Thus, the issue of a low SNR for gradient measurements far away from a node (flat gradients, high noise), is not as problematic as it might seem intuitively.

6.3.2. Wireless Communication

Direct signal strength, i.e., in practice signal-to-noise plus interference (SNIR), measurements are often not possible because commodity radios such as those based on IEEE 802.11 are used in most scenarios. These radios do not allow user access of actual physical parameters such as noise measurements or signal strengths. Nevertheless, they often allow the user to access the so-called RSSI value of each received packet. This value is sufficiently correlated with the real SNIR to use it as an estimator for basic tasks. Actual localization using only this value cam be a rather difficult task. The advantage of using RSSI is that the measured values can uniquely be assigned to a specific network node by the address information of the packet. Information about packages that were only partly received or were not strong enough to be decoded completely are mostly lost.

Most wireless communication technologies work in so called unlicensed ISM bands. Because they are unlicensed, there can be a lot of competition for bandwidth resulting in very space- and time-varying characteristics of network parameters. Thus, algorithms using these parameters have to be very robust. For a more complete discussion on this topic and some examples, refer to section 2.2.2.

Even in free-space, RSSI measurements cannot easily be used for distance estimation. This is due to the nonlinear relationship between RSSI and distance (Goldsmith, 2005). For a general discussion on the propagation of radio waves refer to section 2.3. In realistic scenarios this relationship can also become very noisy because of physical characteristics such as multi-path fading or by interference from external sources. Sophisticated models for instance those used by Fink and Kumar (2010) can partially deal with these problems but they are often computationally costly and rely on accurate odometry and self-localization.

Measuring directions to network nodes is more robust than distance measurements because it requires less information, but it can nevertheless be very noisy, especially if these measurements were derived from RSSI measurements, as done when using estimated gradients for directional information. Thus, the algorithm has to be designed to be able to deal with rather high levels of noise in the direction measurements.

6.4. Exploration Algorithm

As discussed above, only directions to network nodes and no explicit distance estimation are used. The idea of this algorithm is to use RSSI measurements in order to keep some fixed distance to the closest node while circling it until the next node becomes the closest one. If non-line-of sight communication is assumed, the robot follows RSSI isolines. This fixed distance is unknown and does not actually matter for the algorithm as long as it is in some sensible range determined by the mean distance between neighboring nodes of the network and the maximum communication distance. The algorithm is described in algorithm 6.4.1 and a typical run is depicted in fig. 6.1.

Typical values for `lowThreshold` and `highThreshold` are $-65\,\mathrm{dB\,m}$ and $-60\,\mathrm{dB\,m}$ respectively. An abstracted robot inertia $\alpha = 0.1$ (see algorithm 6.4.1) as well as the aforementioned values for the two parameters of the algorithm were chosen for the quantitative analysis.

Algorithm 6.4.1 Network exploration algorithm using direction measurements. The robot is moving with a speed of $\vec{v}$ (here $|\vec{v}| = 1$ for simplicity) and has some inertia abstracted by the use of α (< 1). Network nodes are denoted by $node_i$. Parameters of the algorithm are $lowThreshold$ and $highThreshold$ respectively. The user is required to have a general idea of the location of the network in order to set the robot off into its general direction as a starting point for the algorithm.

Require: move with constant speed $\vec{v}$ in general direction of network
  while not reached starting point **do**
    if some node in communication range **then**
      measure RSSI of visible nodes $\{node_i\}_{vis}$
      choose node with highest RSSI $node_j$
      estimate direction $\vec{d}$ to $node_j$ (with $|\vec{d}| = 1$)
      if RSSI of $node_j < lowThreshold$ **then**
        $\vec{v} \leftarrow (1-\alpha)\vec{v} + \alpha\vec{d}$ ▷ move in direction of $node_j$
      else if RSSI of $node_j > highThreshold$ **then**
        $\vec{v} \leftarrow (1-\alpha)\vec{v} - \alpha\vec{d}$ ▷ move away from direction of $node_j$
      else
        $\vec{q} = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} = \begin{pmatrix} -d_2 \\ d_1 \end{pmatrix}$
        $\vec{v} \leftarrow (1-\alpha)\vec{v} + \alpha\vec{q}$ ▷ move perpendicular to direction of $node_j$ to the left
    move one step with $\vec{v}$

This algorithm exhibits several elegant characteristics. Since the movement is asymmetric, i.e., the robot always moves perpendicular to the left or on the axis of the direction towards the node, but never perpendicular to the right — it can never get stuck. This is the same effect as solving a maze by always keeping to the left wall (Shannon, 1951). Because of the nondeterministic noise, the robot can get stuck in very tight situations for some time due to the noise in the RSSI measurements as shown in fig. 6.2, but ultimately it will always escape again due to noise.

The algorithm operates directly on the noisy direction measurements, which in this simulation are collected by a virtual sensor. In principle, the movement of the robot should therefore also be noisy. This effect is however partly compensated by the inertia of the robot which acts as a kind of mechanical low pass filter smoothing out some of the noise.

6.5. Simulator Details

The simulator used here is strictly two dimensional which is a good approximation of a flying robot at a fixed altitude or a ground-based robot. The robot is simulated as a point-like entity, which is justified since a holonomic robot is assumed and since the length scale of the communication is of orders of magnitudes longer than the size of the robot. The dynamical characteristics of the robot were abstracted using only inertia. Environmental factors such as wind are not simulated.

The main goal for the simulator was to investigate the robustness of the algorithm against noise. Thus the execution of the algorithm is stochastic necessitating a large number of sim-

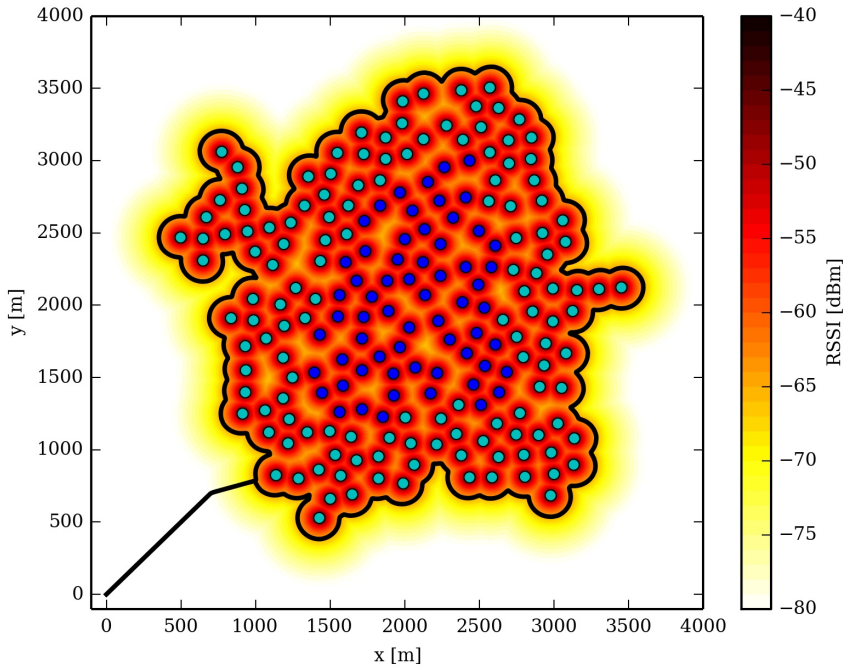


Figure 6.1.: A typical run of the algorithm for a typical node placement is depicted. The nodes are shown as dots, their color denotes if they have been visited or not. Cyan nodes were in communication range of the robot at some time during the execution of the algorithm and blue nodes were not. Additionally, the maximum RSSI value from all nodes is shown for every point as the underlying color plot. The black line is the path the robot took in this run and nicely follows an RSSI isoline. In the shown simulation the measurements of the robot were noise-free.

ulations to be able to make statistically sound statements. This means that the performance of the simulator has to be reasonably high, which is why the so called simplified path loss model (see section 2.3.1) with an exponent of 3 (Goldsmith, 2005) was used instead of more complex signal propagation methods. Some of these more complex — and therefore more computationally costly — models were also tested but yielded qualitatively identical results. Additionally, signal propagation models for a flying robot are in general more simple because they often operate under line-of-sight conditions.

In order to keep the simulator as simple as possible, and according to the intended use for flying or ground-based robots in an open space, no environment was simulated. This avoids having to deal with obstacle avoidance and more complex issues such as path planning. Furthermore, having to rely on obstacle avoidance implies non-line-of-sight conditions which in turn would necessitate much more complex signal propagation models which do not change the results qualitatively.

Noise was simulated by simply adding a random variable with a Gaussian distribution to the virtual sensors of the robot. The random number generation was carried out by Pythons

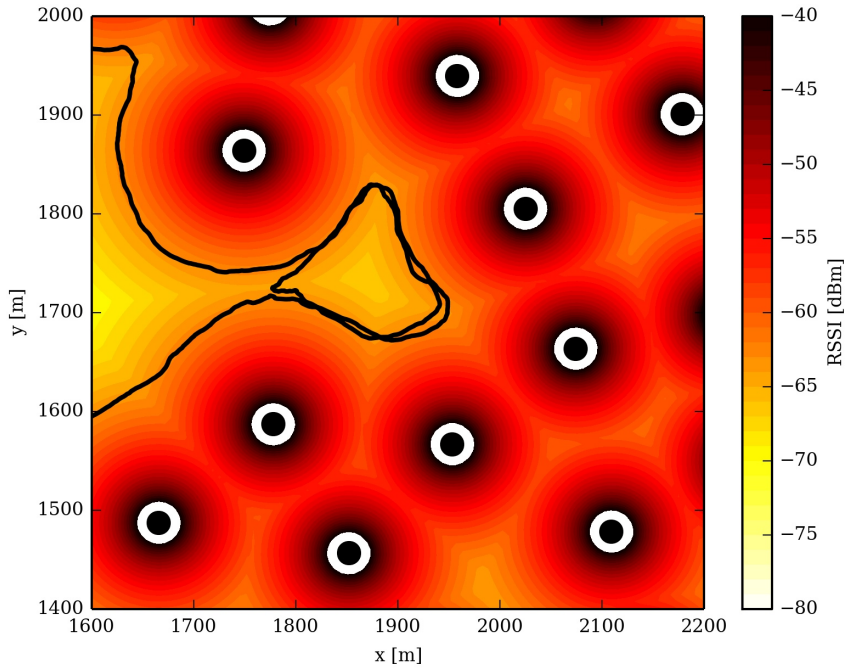


Figure 6.2.: A situation where the robot got stuck between several nodes, forming some kind of cavity inside of the network, is shown. As can be seen, the robot performs one additional turn before it can escape this situation. This is a totally random process as it is mainly due to the noise in RSSI measurements. The black line is the path the robot took, the nodes are shown as black dots and the maximal RSSI value from all nodes is shown for every point as the underlying color plot.

random package[3], i.e., by a Mersenne Twister.

The nodes where placed in a random manner while guaranteeing a connected network. The algorithm was inspired by NPART (Milic and Malek, 2009). This algorithm was modified to get a more uniform distribution of the nodes more typical of sensor or mesh networks. Basically, the algorithm starts with a randomly placed node. Then it picks a node from the existing nodes and places a new node at a fixed distance in a random direction from this node in a way so that the new node is not closer than this distance to any other node. This process is repeated until all nodes are placed. A typical distribution is depicted in fig. 6.1.

The simulated area was $4\,\mathrm{km}$ by $4\,\mathrm{km}$ with 200 simulated nodes with a communication range of $150\,\mathrm{m}$. The simulator was implemented in Python using the standard packages NumPy/Scipy (Jones et al., 01) and multiprocessing[4]. Visualization was done using the matplotlib (Hunter, 2007) package.

[3] `https://docs.python.org/2/library/random.html`
[4] `https://docs.python.org/2/library/multiprocessing.html`

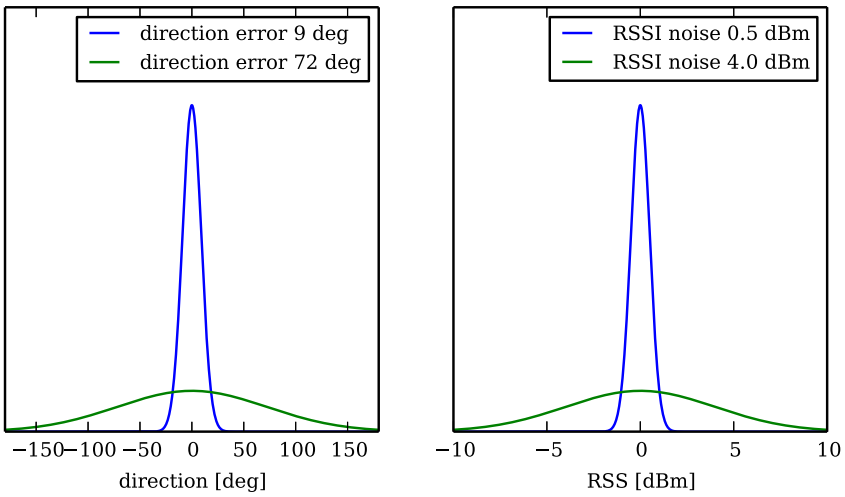


Figure 6.3.: Shown here are the two most extreme noise distributions from the parameter study for the two types of noise respectively. Remarkable is the most noisy distribution for the direction measurements which spreads from $-150°$ to $150°$ rendering a measured direction almost devoid of information.

6.6. Simulation Results

The algorithm takes two kinds of measurements, namely directions to the nodes in communication range and corresponding RSSI values. The noise is modeled on both sensory inputs as Gaussian with zero mean and some standard deviation. Since the units of RSSI are in dB, the measurements are logarithmic so the noise is actually log-normal according to theory. The noise on the direction measurement acts directly on the relative bearing angle between the robot and the network node for which the direction is measured.

A parameter study in the standard deviations of the two kinds of noise was performed. For the noise of the direction an interval from approximately $9°$ to $72°$ was chosen. The RSSI noise spanned an interval from $0.5\,\mathrm{dB}$ to $4\,\mathrm{dB}$. Both intervals were sampled at 8 positions yielding 64 different noise configurations. For both kinds of noise the two most extreme cases are depicted in fig. 6.3 in order to gain a graphical intuition of the noise intervals used.

6.6.1. Effectiveness of the Algorithm

For all runs the outline was verified to be a closed loop, which is a measure of the effectiveness of the algorithm. This means that a closed loop is counted as a success and one which isn't as a failure, since a closed loop implies that the network was outlined correctly. This is true because the network in simulation is known to be finite and by construction connected. The initial movement towards the network, which can be seen as a straight line in fig. 6.1, was not taken into account in order to only assess the real outline of the exploration algorithm.

During the parameter study 1000 simulations were performed for each noise configuration in order to be able to make statistically sound statements. Thus a total of 64000 simulation

runs was performed. Remarkably, every one of these simulations resulted in a closed loop independent of the amount of noise.

Examination of results of single runs showed that the robot can get stuck in narrow cavities between nodes as discussed above and depicted in fig. 6.2. This figure shows a rather short episode where the robot does one additional loop before being able to escape. Depending on the actual node placement and noise realizations, the robot can get stuck for an extended period of time going in circles. Nevertheless, in every single of the simulation runs, the robot could escape after some time. This is due to the fact that the probability for the robot entering a loop due to noise is the same as the probability for exiting it due to noise.

6.6.2. Robustness of the Algorithm

Node placement as well as both noise components are random, allowing statistical statements about robustness. 1000 simulation runs were conducted per noise configuration and for each run, i.e., for each node configuration, the algorithm was executed once without noise as a baseline and once with noise in order to study the effect of noise. Both runs had identical initial conditions.

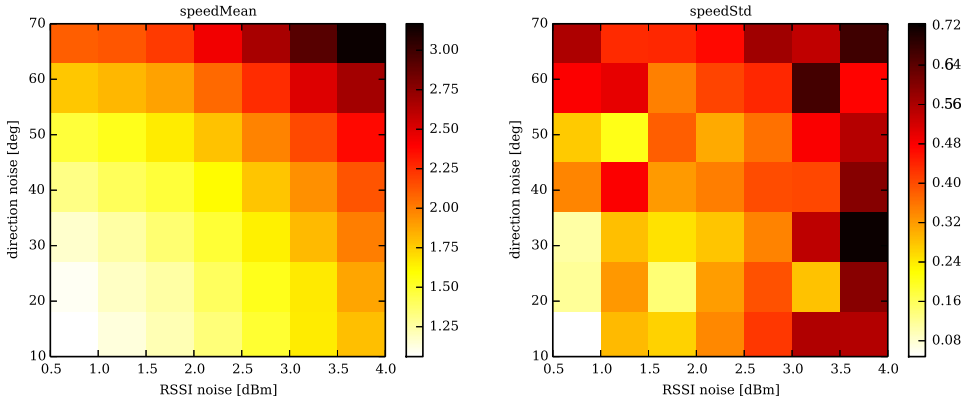
As metrics, the number of nodes which were in communication range of the robot at least once during the execution of the algorithm, (from now on called visited nodes), and the time it took the robot to complete one loop (the speed was constant for all runs) in relation to the respective numbers generated by the run without noise was chosen as a metric. Thus, the two metrics are fractions of the number of visited nodes and relative algorithm execution time.

The results of these simulations are depicted in fig. 6.4. The plots show mean and standard deviation of both metrics, respectively.

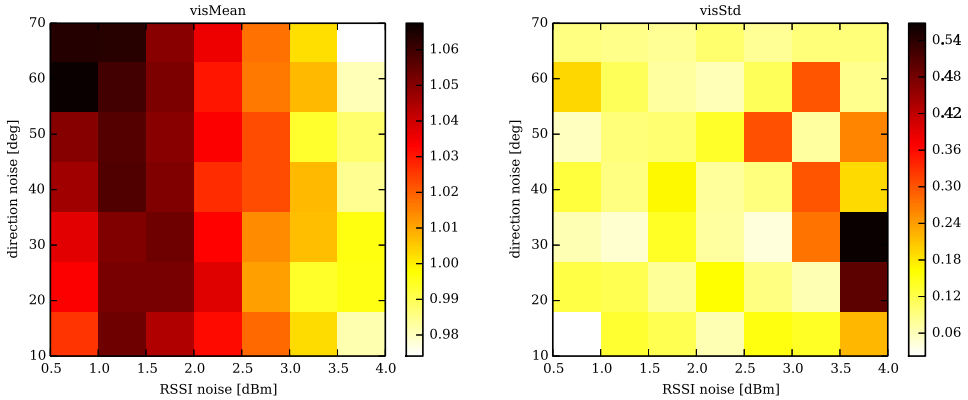
The relative execution time of the algorithm increases as a function of noise and the increase is independent of the type of noise. The standard deviation shows roughly the same behavior. This is to be expected since more noise in the sensory input also means more noise in the movement of the robot, leading to more jitter and a longer effective traveled distance.

The fraction of visited nodes shows a more interesting behavior. This metric is largely independent of the direction noise and only a function of noise in the RSSI measurements. The standard deviation seems to be rather independent of both. For low RSSI noise levels this metric is even greater than 1.0, meaning that more nodes were discovered than the noise-free run did. For larger noise levels, this metric again drops below 1.0. There seems to be some optimal small noise which in a way encourages exploration while degrading speed only in a minimal way.

The amount of noise the algorithm can tolerate is remarkable. A standard deviation of 72° in the direction measurement means that very little information is actually transported in the sensory data (as can be seen in fig. 6.3). The results further suggest that even higher levels of RSSI noise could be tolerated. This result is interesting for gradient-based direction measurements since those show low SNR for measurements far away from the source because the gradients become very flat, which is due to the propagation characteristics of electromagnetic waves (for details refer to section 2.3), but the noise levels (due to external interference etc.) stay constant.



(a) Execution time metric of the simulation data. Depicted are mean and standard deviation.



(b) Fraction of visited nodes metric of the simulation data. Depicted are mean and standard deviation.

Figure 6.4.: Results of the parameter study simulations. Depicted are the two metrics: relative execution time in fig. 6.4a and fraction of visited nodes in fig. 6.4b with their mean and standard deviation respectively as a function of the two kinds of noise. For each of the noise configurations (direction bearing angle noise and RSSI noise) 1000 simulations were conducted as basis for each distribution.

6.7. Summary and Outlook

An algorithm to explore the outline of a wireless network for a mobile robot and a minimal simulation framework in which this algorithm was evaluated, was presented. A parameter study was performed in simulation and the expected characteristics of the algorithm were confirmed.

6.7.1. Conclusion

The algorithm is, as expected, very robust against noise. The performance naturally degrades with increasing noise but its effectiveness is not affected. Small amounts of noise in RSSI

measurements, which control the distance to the network nodes, can even be of benefit as they increase the exploratory behavior of the robot.

This very basic and computationally cheap algorithm can cope with very noisy measurements and work with minimal information. Neither complex distance estimations nor odometry or self-localization are necessary to complete this task. Algorithms such as the one presented which try to work with as little explicit information as possible are well suited for a future world with cheap autonomous robotic applications.

6.7.2. Future Work

The next step, building on this algorithm, will be to actually localize network nodes. By exploiting the sensorimotor interaction, this may be achieved without complex localization techniques via RSSI measurements, but by using the directional data gathered while outlining the network. Combining this information with odometry of some kind could facilitate some rough localization of network nodes via some sort of triangulation which can be precise enough for a number of applications.

Taking more realistic physical scenarios like non-line-of-sight and obstacles into account and perhaps even exploiting these effects could be another direction for future work. This also includes robots in more confined spaces for instance on a grid of streets in an urban scenario.

This work was presented in (Blum and Hafner, 2014)[1].

Chapter 7

Gradient-Based Taxis Algorithms for Network Robotics

7.1. Introduction

Finding the physical location of a specific network node is a prototypical task for navigation inside a wireless network. Here, the implications of wireless communication as a measurement input of gradient-based taxis algorithms are considered in depth. The question of how gradients can be measured is discussed and the errors of this estimation are determined. Then, a gradient-based taxis algorithm is introduced as an example of a family of gradient-based, convergent algorithms and its convergence in the context of network robotics is discussed. An exemplary experiment to illustrate how to overcome some of the specific problems related to network robotics is conducted. Finally, it is shown how to adapt this framework to more complex objectives.

7.2. Problem Statement

In this chapter, a very basic but nevertheless very relevant task for a robot is considered: finding the physical location of a specific network node is the prototypical task for many tasks dealing with navigation inside a wireless network. Later it will be also shown how the framework resulting from this basic task can be readily adapted to cope with more complex objectives.

For this framework a robot is assumed to receive network packets from the network node to be found at different spatial positions. It then measures the signal strength of these packets as a scalar measurement. Thus conceptually, a scalar field is sampled at different points in space. Furthermore, the robot does not have any means to detect the direction from which the packets arrive as would be possible with a directional antenna. Additionally, it cannot measure the time-of-flight of the packets, which would enable the robot to simply trilaterate the source. While both options are technologically feasible, they are not considered here in order to be independent of additional technology and hardware as would be needed for such time-of-flight or directional measurements.

[1]For a detailed breakdown of the contributions of the different authors refer to section 1.5.

In addition to measurements of the signal strength, the robot is able to perform local odometry measurements. They are local in the sense that they are precise for small distances but not sufficient for global localization by, for example, path integration of these measurements. Other global localization methods such as GPS measurements are not available to the robot.

For the sake of simplicity an obstacle-free space is assumed, which is valid in some scenarios, for example with flying robots or ground-based robots in open space. In other scenarios the algorithm has to be expanded by obstacle avoidance techniques for which there exist off-the-shelf methods that are not the focus of the discussion. The algorithm presented here converges from any starting point, thus obstacle avoidance can be implemented, for example by stopping the taxis algorithm and restarting it after clearing the obstacle.

Therefore, this chapter focuses mainly on the gradient estimation and its properties in the context of wireless networks. The implication of this use case of network robotics on the gradient estimation and specifically the implications on the convergence of the taxis algorithm are discussed in detail. In section 7.3 the nature of wireless communication with particular attention on the proposed algorithms is discussed. Then different noise sources and their effect on the estimated gradients are discussed in section 7.4. In section 7.5 these gradients are used to introduce a stochastic approximation algorithm and translate it into the robotics world. The convergence results of this algorithm for the case of wireless communication can then be extended to the whole family of stochastic approximation algorithms. An exemplary implementation of one of this algorithms is presented in section 7.6 which shows how to deal with some of the specific challenges posed by this case of network robotics.

After discussing the properties of the gradient estimation and the convergence of the algorithm for the case of pure signal strength measurements in the context of network robotics, as a last step, it is presented in section 7.7 how this proposed algorithm can be adapted to more advanced tasks in the same framework.

7.3. Physical Propagation Model

The underlying physical models for path loss are considered in depth in this section. This review forms the basis for the discussion of the algorithm itself and is of universal importance for all algorithms in network robotics. This section focuses on the characteristics of wireless communication specifically from the point of view of convergence conditions of the algorithm. For a more complete and general discussion, refer to chapter 2.

7.3.1. Minimalistic Physical Propagation Model

The measured signal strength depends on a lot of environmental parameters and is in general computationally expensive to calculate. In many cases, realistic signal strength values can only be obtained through direct measurement or numerical solution of Maxwell's Equations (see section 2.1). A minimalistic model of path loss, which ultimately defines the signal strength measured by a user, is a simplified path loss model derived from the open space path loss model. It can be written as (Goldsmith, 2005, Chap. 2)

$$P(x) = -10\gamma \log_{10}\left(\frac{x}{d_0}\right) \tag{7.3.1}$$

in dB where d_0 and γ are empirical constants and x is the distance between receiver and transmitter. d_0 has typical values of several meters and γ is about 3 for urban environments.

Since this model is only used to get an intuition about the general behavior of path loss and thus signal strength, for simplicity only the one-dimensional rotation-symmetric version of path loss is used. To get an intuition about its derivatives (and thus also its gradients) the first three derivatives of the path loss are proportional to

$$\partial_x P(x) \propto -\frac{1}{x}, \tag{7.3.2}$$

$$\partial_x^2 P(x) \propto \frac{1}{x^2}, \tag{7.3.3}$$

$$\partial_x^3 P(x) \propto -\frac{1}{x^3}. \tag{7.3.4}$$

This model can be motivated as physically plausible for the case of free space with some effective signal attenuation.

7.3.2. Elaborate Physical Propagation Model

In general, Maxwell's equations (see section 2.1) have to be solved to correctly calculate radio wave propagation. With complete knowledge of the environment this is indeed possible but in practice almost never feasible due to the high computational costs and as complete knowledge of the environment is unobtainable in real scenarios. For a more complete discussion on the feasibility of complete simulation of Maxwell's equations and the requirements on hardware and environmental knowledge for a realistic example, refer to section 2.1.3.

As a consequence, effective radio propagation models have been developed, which in general consist of three main components. These three components are path loss (large scale), shadowing (medium scale) and fading (small scale) (Goldsmith, 2005). In the following paragraphs these components are discussed with a robot as a network node in mind.

Large scale path loss can be identified with the simplified path loss model discussed in section 7.3.1. The most straightforward case for this is the free space model which takes only the most basic physical effect into account, i.e., wave propagation in free space. More elaborate models usually approximate all kinds of empirical effects on path loss by adjusting the exponent of the path loss function eq. (7.3.1). This exponent is tuned to different scenarios via empirical measurements. For the following discussion on taxis algorithms, it is important to note that these models are strictly monotonically decreasing.

Shadowing is the effect of large obstructions such as a hill or wall in the direct path of wave propagation. In some cases, for instance the standard double plasterboard wall in an office environment, these effects can simply be measured and added as additional path loss. More complex scenarios need more complicated models. Note that shadowing can only attenuate signals.

Some configurations of obstructions, consisting for example of walls of different attenuation, can create situations in which a robot simply following the proposed algorithms can get stuck because the signal strength gradient would try to guide the robot through a wall. An obstacle avoidance algorithm, which needs some degree of knowledge about the environment, has to be used to help the robot escape these situations. As stated earlier, the taxis algorithm itself converges regardless of the starting point so it can be stopped when the robot starts the obstacle-avoidance algorithm and restarted once it has cleared the obstacle.

Finally, fading is a result of multi-path propagation of radio waves. Superposition of waves traveling different paths interfere because of the different phases of these waves on a physical level either constructively or destructively. This leads to spatially varying signal strengths on the length scale of the wave length of the radio signal (about $12.5\,\mathrm{cm}$ for $2.4\,\mathrm{GHz}$). Fading will be discussed in detail in section 7.3.3.

Taking these three effects together, a robot measuring the signal strength of a fixed receiver deals with a strictly monotonically increasing function with a considerable amount of noise and spatially varying characteristics.

Integrating over the different angles, the signal strength is in general — because of shadowing and fading — a non-monotonic function of distance to the transmitter, preventing direct distance estimations using only signal strength measurements.

For moving robots, Doppler shift (Goldsmith, 2005) can be an additional factor. In general, Doppler shift does not affect signal strength on a global level. However, for a sender-receiver pair working in some specified frequency band, Doppler shift can lead to a reduced received signal strength by shifting part of the signal to frequencies which are outside of the communication channel bandwidth. Doppler shift is dependent upon variables such as the speed of the robot, carrier frequency and modulation. For typical robot speeds and commonly used IEEE 802.11 based $2.4\,\mathrm{GHz}$ communication, the effect is negligible ad the discussion in section 2.3.2 shows. Additionally, most robots are capable of holding their position — for example hovering flying robots — for the time it takes to make a measurement mitigating Doppler shift completely. This works trivially for ground based robots but also for some flying robots but is an issue for flying robots based on the fixed wing principle.

7.3.3. Small Scale Fading

As discussed in the previous section, small scale fading is the result of interference and has an effect on the length scale of the wavelength. It leads to fluctuations with periodic character with a periodicity of the wavelength of the wireless signal (which is dependent on the used channel, i.e., the carrier frequency). This effect is deterministic and can lead to local minima in the signal strength, which would violate some convergence conditions (see section 7.5.2).

An example measurement is depicted in fig. 7.1. The deterministic fluctuations on the scale of the wavelength ($12.5\,\mathrm{cm}$ for $2.4\,\mathrm{GHz}$ in this case) can clearly be seen. These variations can have a higher amplitude than than the noise of the signal and have to be dealt with in order to ensure convergence of the presented algorithms.

For a flying robot, this is typically not a problem because it cannot hold position with a precision of this magnitude. This means that because of the stochastic movement (induced by aerodynamics and external factors like wind) these deterministic fluctuations are turned

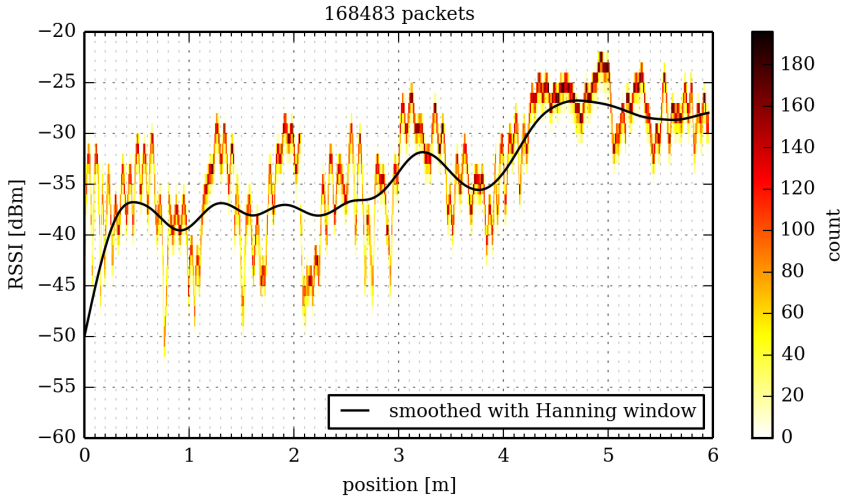


Figure 7.1.: Example indoor measurement of small scale fading caused by walls and other scatterers. The distribution of the signal strength of the measured packages as well as smoothed version of the data is depicted.

into additional noise. For ground based robots, this effect has to be mitigated, for example by taking several samples and averaging over an area larger than the wavelength. This strategy will also be demonstrated in the experiments in section 7.6.

For more exemplary measurements, especially more examples of small scale fading, refer to chapter 5, specifically to section 5.3.

7.3.4. Abstract Propagation Model: Notation

Based on the more elaborate models discussed above, an abstract model is described to be used for a compact notation in the discussion on gradient estimation and convergence. It encompasses all previously discussed more complex models and is deliberately imprecise to enable the short notation. It should be noted that it implicitly contains all characteristics needed for the convergence discussion.

In the model, the true signal strength $f(\vec{x})$ can only be measured up to a measurement error

$$f_{\text{mes}}(\vec{x}) = f(\vec{x}) + \epsilon(\vec{x}) \tag{7.3.5}$$

where the measurement error $\epsilon(\vec{x})$ is a random variable with

$$E[\epsilon(\vec{x})] = 0, \tag{7.3.6}$$
$$V[\epsilon(\vec{x})] = \sigma^2. \tag{7.3.7}$$

The measurement errors $\epsilon(\vec{x})$ are i.i.d. for the individual measurements. Two facts should be noted: first this definition does not make any assumptions about the distribution as long

as the two conditions above are met, and second this condition can be violated by small scale fading (see section 7.3.3) if not dealt with correctly. The signal strength function $f(\vec{x})$ is a scalar field with one global maximum. This model is unspecific about the actual characteristics of the function $f(\vec{x})$. When discussing the convergence properties of the proposed algorithm in section 7.5.2, its characteristics are discussed in detail. For now it serves as a purely notational convenience.

7.4. Gradient Estimation

7.4.1. Central Differences

The true gradient

$$\vec{g}(\vec{x}) = \nabla f(\vec{x}) \tag{7.4.1}$$

is estimated using central differences (ith component):

$$\hat{\vec{g}}_i(\vec{x}) = \frac{f(\vec{x} + h\vec{e}_i) - f(\vec{x} - h\vec{e}_i)}{2h}, \tag{7.4.2}$$

where $\hat{\vec{g}}(\vec{x})$ denotes the estimate of the true gradient $\vec{g}(\vec{x})$, $\vec{e}_i$ is the ith unit vector and h is the used stepwidth. This estimation is the most standard one with improved precision in relation to one-sided finite differences. Right now this estimation does not take into account the measurement errors, which will be discussed in section 7.4.2, but only the errors from the numerical approximation.

Using second order Taylor expansions (Bronstein and Semendjajew, 2008)

$$f(\vec{x} \pm h\vec{e}_i) = f(\vec{x}) + \epsilon(\vec{x}) \pm h\partial_{x_i} f(\vec{x}) + \frac{h^2}{2}\partial^2_{x_i} f(\vec{x}) \pm \frac{h^3}{6}\partial^3_{x_i} f(\vec{x} \pm \xi h\vec{e}_i) \tag{7.4.3}$$

with $0 < \xi < 1$ yields for the gradient estimate

$$\hat{\vec{g}}_i(\vec{x}) = \partial_{x_i} f(\vec{x}) + \frac{h^2}{12}\left(\partial^3_{x_i} f(\vec{x} + \xi_1 h\vec{e}_i) - \partial^3_{x_i} f(\vec{x} - \xi_2 h\vec{e}_i)\right) \tag{7.4.4}$$

where $0 < \xi_1 < 1$, $0 < \xi_2 < 1$.

7.4.2. Error Analysis

A gradient estimate calculated with measured data contains, additionally to the numerical errors discussed above, measurement errors. The notation $f_{\text{mes}}(\vec{x}) = f(\vec{x}) + \epsilon(\vec{x})$ is used as described in the abstract model in section 7.3.4 to write the estimated gradient based on measurements $\hat{\vec{g}}_{\text{mes}}(\vec{x})$ as

$$\begin{aligned} \hat{\vec{g}}_{\text{mes},i}(\vec{x}) &= \frac{f_{\text{mes}}(\vec{x} + h\vec{e}_i) - f_{\text{mes}}(\vec{x} - h\vec{e}_i)}{2h} \\ &= \hat{\vec{g}}_i(\vec{x}) + \frac{\epsilon(\vec{x} + h\vec{e}_i) - \epsilon(\vec{x} - h\vec{e}_i)}{2h}. \end{aligned} \tag{7.4.5}$$

The expectation value and variance of $\hat{\vec{g}}_{\text{mes}}$ can now be calculated as

$$E[\hat{\vec{g}}_{\text{mes},i}(\vec{x})] = \partial_{x_i} f(\vec{x}) + \frac{h^2}{12}\left(\partial^3_{x_i} f(\vec{x} + \xi_1 h\vec{e}_i) - \partial^3_{x_i} f(\vec{x} - \xi_2 h\vec{e}_i)\right), \tag{7.4.6}$$

$$V[\hat{\vec{g}}_{\text{mes},i}(\vec{x})] = \frac{\sigma^2}{2h^2}. \tag{7.4.7}$$

As expected, the relation $E[\hat{\vec{g}}_{\text{mes}}(\vec{x})] = \vec{g}(\vec{x}) + O(h^2)$ holds for the expectation value of the estimated gradient. Estimating the gradient using central differences without measurement errors yields the same relation for the expectation value.

There are two kinds of errors contained in $\hat{\vec{g}}_{\text{mes}}(\vec{x})$, namely a numerical error produced using finite differences which behaves like $O(h^2)$ and a stochastic error due to the measurement error $\epsilon(\vec{x})$ which behaves like $O(h^{-2})$.

7.4.3. Motor Noise: Measurement Errors

For a real robot, there is, in addition to the physical measurement noise ϵ as in eq. (7.3.5), so called motor noise. This noise is added to any motor command and thus affects all movements. In general this noise is vectorial and affects every infinitesimally small movement of the robot thus leading to a random walk-like behavior. This behavior is depicted in the inset of fig. 7.2 in a graphical way.

Since the argument is purely statistical and no fixed coordinate system is used, the actual end position of the robot is not important. Thus, abstracting here from this vectorial noise to a noise in the direction of movement only does not make a qualitative difference for the argument, but does simplify notation. For the central differences this means that $f(\vec{x})$ is not sampled at $f(\vec{x} + h\vec{e}_i))$ but at $f(\vec{x} + (h + \epsilon_h)\vec{e}_i))$. This yields a gradient estimation of

$$\hat{\vec{g}}_{\text{mes},i}(\vec{x}) = \frac{1}{2h}(f_{\text{mes}}(\vec{x} + (h + \epsilon_{h,1})\vec{e}_i) - f_{\text{mes}}(\vec{x} - (h + \epsilon_{h,2})) \tag{7.4.8}$$

The ϵ_h are assumed to be i.i.d. and bias-free. The specific distribution has no influence on the results of the discussion here. Even a bias in the distribution, for example the robot always moving more to the left, has no influence since in the algorithm, the movement direction itself is a stochastic quantity. In contrast to a real random walk, the end positions are only distributed with a one-dimensional distribution in the direction of movement. Writing the full vectorial distributions would clutter the formulas but not change the results so only the simplified case is discussed here.

For small ϵ_h the $f(\vec{x} + (h + \epsilon_h)\vec{e}_i)$ can be expanded with

$$f(\vec{x} + (h + \epsilon_h)\vec{e}_i) = f(\vec{x} + h\vec{e}_i) + \epsilon_h \partial_{x_i} f(\vec{x} + h\vec{e}_i) + O(\epsilon_h^2). \tag{7.4.9}$$

For small ϵ_h this expansion is valid and effectively yields a larger measurement error of $f(\vec{x})$ than the pure physical measurement error ϵ. For further analysis this additional error can be absorbed in the measurement error as long as their distributions are similar. In general, the central limit theorem (Blobel and Lohrmann, 1998, p. 129) holds and allows us to simply add both errors. Note that while the path loss noise is often modeled by Rayleigh

or Rician fading models, which result in non-Gaussian noise, these models are stochastic approximations to multi-path fading. Multi-path fading itself however is deterministic and only the measurement errors themselves are stochastic and Gaussian (see also section 7.3.3 and section 2.3.2)

7.4.4. Motor Noise: Iteration Steps

In addition to the effect of motor noise discussed in section 7.4.3, the error in the actual movement of the robot when iterating the algorithm has to be considered.

Motor noise in the movement of the robot — as long as it is truly random and not biased — can be thought of as an additional error in the estimated (or in a way measured) gradient of $f(\vec{x})$. This discussion is similar to the one in section 7.4.3.

This error is constant per unit length but because the stepwidth a_k is decreasing (see section 7.5.2), the resulting total movement error is also decreasing. Since the error due to the central difference approximation eq. (7.4.7) increases with decreasing h_k, the movement error becomes insignificant for large k, thus not influencing convergence.

7.4.5. Signal-To-Noise Ratio

The SNR is a useful quantity describing the relation of a measured signal amplitude to the amplitude of the noise introduced into this channel. In the case of gradient estimation it can be used to characterize the quality of the estimated gradients in terms of a (virtual) sensor reading.

Defining the SNR of some function $e(\vec{x})$ as (Smith, 1997, Chap. 2)

$$SNR = \frac{E[e(\vec{x})]}{\sqrt{V[e(\vec{x})]}} \tag{7.4.10}$$

and applying this definition for the ith component of the estimated gradient $\hat{\vec{g}}_{\text{mes},i}(\vec{x})$ yields

$$SNR_i = \frac{\sqrt{2}h}{\sigma}\partial_{x_i} f(\vec{x}) + \frac{\sqrt{2}h^3}{12\sigma}\left(\partial^3_{x_i} f(\vec{x} + \xi_1 h\vec{e}_i) - \partial^3_{x_i} f(\vec{x} - \xi_2 h\vec{e}_i)\right) \tag{7.4.11}$$

which can be approximated for small h with

$$SNR_i \sim \frac{h}{\sigma}\,\partial_{x_i} f(\vec{x}). \tag{7.4.12}$$

For large h the error due to the remainder term of the Taylor expansion, which is the bias of the gradient estimation, dominates. The SNR as defined above then loses its meaning.

For the specific problem of path loss eq. (7.3.1) as the function for which the gradient is estimated, the third derivative of the path loss eq. (7.3.4) vanishes like $\frac{1}{x^3}$. This means that far from the source this approximation is valid also for large h since the bias only increases like h^2.

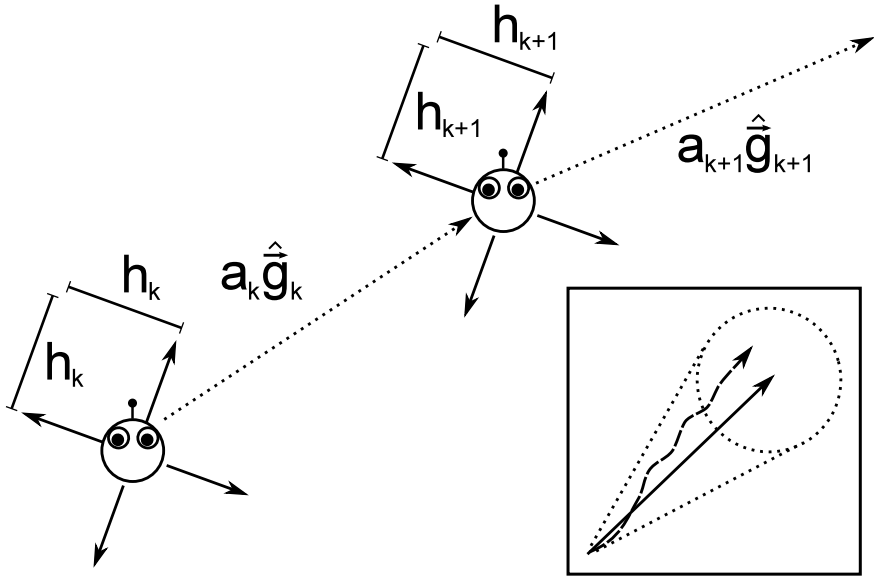


Figure 7.2.: The sequence of two iterations. The arrows denoted with lengths of h_k designate the steps taken for gradient estimation. The dotted arrows denoted with a_k show the actual iteration steps. The inset illustrates the effect of motor noise. The step to be taken is depicted as a solid arrow, the approximate distribution of end position with a dotted circle and one of the realizations of movement with motor noise as a dashed arrow.

7.5. Taxis Algorithm

The robot estimates gradients using finite differences by sampling the scalar field and uses this gradient as the direction for the next step. The algorithm is formally introduced and its convergence and statistical properties are discussed.

7.5.1. Finite Difference Stochastic Approximation

One of the most basic algorithms to find the minimum of some scalar field is the method of steepest descent. It consists of calculating (or in this case estimating) the gradient $\hat{\vec{g}}_{\text{mes}}(\vec{x})$ beginning at some starting point $\hat{\vec{x}}_0$ and following it with some stepwidth a_k:

$$\hat{\vec{x}}_{k+1} = \hat{\vec{x}}_k + a_k \hat{\vec{g}}_{\text{mes}}(\hat{\vec{x}}_k). \tag{7.5.1}$$

Additionally, the stepwidths h of the gradient estimation eq. (7.4.2) may also be adapted for each step and are thus denoted as h_k. The iterates of this algorithm are estimates of the position of the minimum, starting from an initial guess $\hat{\vec{x}}_0$, which is in this case trivially the position of the robot when the algorithm is started, and are therefore denoted as $\hat{\vec{x}}_k$. Thus, this algorithm is basically stateless and the current position of the robot is always the best estimate of the minimum. The gradients then always point the way in which the robot has

to move to reach the minimum. Two iteration steps of this algorithm are depicted in fig. 7.2 in a graphical way.

This algorithm has been subject to theoretical and numerical analysis since the 1950s because it can be used to solve common optimization tasks in many scientific and engineering areas. This particular method based on central differences is known as Finite Difference Stochastic Approximation (FDSA) in the field of Stochastic Approximation (Spall, 2005).

7.5.2. Convergence

It can be shown that this algorithm converges if a_k, h_k, $f(\vec{x})$ and $\epsilon(\vec{x})$ conform to some conditions (Spall, 2005, pp. 159-162). Informally speaking these conditions demand that

- $f(\vec{x})$ has a global minimum at $\vec{x}^\star$,
- $\epsilon(\vec{x})$ has a mean of zero and finite variance,
- the Hessian Matrix $H(\vec{x}) = \frac{\partial^2 f(\vec{x})}{\partial \vec{x} \partial \vec{x}^\top}$ exists and is uniformly bounded for all $\vec{x}$.

The first condition is discussed in section 7.3.2 and is found to be satisfied for the case of signal strength as the scalar field from which the samples used to estimate the gradient are taken. It has one global maximum which can easily be turned into a global minimum by multiplying the measured values by -1.

The second condition constrains the noise of the signal. Measurement noise as modeled in section 7.3.4 has mainly physical origins and is discussed in section 7.3.2. Additionally, there is noise originating from the motors of the robot as discussed in section 7.4.3. Finite variance of the noise is easily satisfied by all real systems. Small scale fading can be a deterministic bias for the signal strength measurements as discussed in section 7.3.3, but that bias can be dealt with as illustrated in section 7.6. The bias of motor noise has been discussed to be zero in section 7.4.3 because of the stochastic nature of the iterate.

The third condition — in simplified terms — requires the scalar field to be smooth. This constraint is satisfied because path loss is a physical effect and physical fields governed by Maxwell's equations always fulfill this smoothness condition.

Furthermore (and with the most practical relevance), gain sequences a_k as well as the sequences of step sizes h_k for the central differences are restricted for a convergent algorithm:

$$a_k > 0, \quad h_k > 0, \quad a_k \to 0, \quad h_k \to 0, \tag{7.5.2}$$

and

$$\sum_{k=0}^{\infty} a_k = \infty, \quad \sum_{k=0}^{\infty} a_k h_k < \infty, \quad \sum_{k=0}^{\infty} \frac{a_k^2}{h_k^2} < \infty. \tag{7.5.3}$$

Thus, $h_k \to 0$ slower than a_k. These conditions have to be satisfied by a practical implementation of this algorithm. However, these cannot be considered as design guidelines since they only restrict the design space of the algorithm. The next section can give more insight into the choice of parameter sequences for a_k and h_k.

7.5.3. Distribution of the Iterate

Unfortunately, there is no known finite-sample ($k < \infty$) distribution for $\hat{\vec{x}}_k$ for general non-linear problems (Spall, 2005, p. 112). But asymptotic ($k \to \infty$) normality of this distribution can be shown for more specific choices of a_k and h_k (Spall, 2005, p. 162-164):

$$a_k = \frac{a}{(k+1+A)^\alpha}, \tag{7.5.4}$$

$$h_k = \frac{h}{(k+1)^\gamma}. \tag{7.5.5}$$

Here $a > 0$, $h > 0$, $\alpha > 0$, $\gamma > 0$ is assumed. $A \geq 0$ is a stability constant which ensures small enough gains in the beginning and large enough gains in the end.

In order to show asymptotic normality, some constraints on these constants have to be added. The most practically relevant ones are:

$$\beta \equiv \alpha - 2\gamma > 0, \tag{7.5.6}$$

$$3\gamma - \frac{\alpha}{2} \geq 0. \tag{7.5.7}$$

If these conditions are satisfied, asymptotic normality of $\hat{\vec{x}}_k$ can be shown. The forms of the mean and variance of the resulting Gaussian distribution are unwieldy but closed-form expressions of both can be found in the literature.

The rate of stochastic convergence of $\hat{\vec{x}}_k$ to $\vec{x}^\star$ is then proportional to $k^{-\frac{\beta}{2}}$. β is maximized at $\alpha = 1$ and $\gamma = \frac{1}{6}$ leading to a maximal attainable stochastic convergence rate of $k^{-\frac{1}{3}}$. This can in turn serve as a general guideline for the choice of the parameter sequences. Further details and design guides for the practical choice of the series can be found in Spall (2005).

In principle the particular choice of the series a_k and h_k weighs exploration against exploitation. These parameter series do not have to be analytic. Consequently, they can be adaptive as long as the conditions discussed in section 7.5.2 are met. With this step ideas from, for example, infotaxis (Moraud and Martinez, 2010) can be incorporated.

7.5.4. Other Stochastic Approximation Algorithms

FDSA is only one algorithm of a family of stochastic approximation algorithms which have similar characteristics and similar convergence conditions. It was chosen because of the intuitive formulation of the gradient estimation which allows discussion of the properties of the estimated gradients more clearly in the case of network robotics.

Other algorithms from this family work with different formulations for the gradient estimation and can be proved analytically to converge. They are known from the stochastic approximation literature (Spall, 2005) and have been used successfully in robotics contexts (Atanasov et al., 2012).

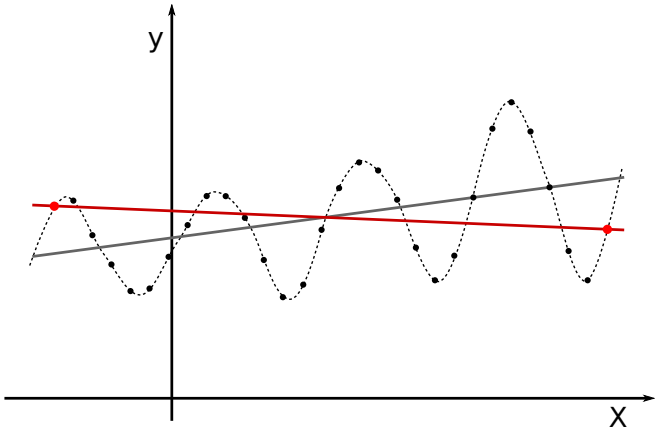


Figure 7.3.: Schematic example of mitigating the effects of fading. A typical fading signal is sampled noise-free only at the two red points such as done by pure finite differences leading to the wrong red linear model. Sampling along the signal for longer than its wavelength as done by the black sample points however leads to the correct large scale linear model shown in dark gray.

Similar approaches to taxis problems based on stochastic approximation have been proposed (Atanasov et al., 2012)[2]. These algorithms make use of more efficient gradient estimations using RDSA which needs less samples from the measured field than FDSA while offering the same performance.

Another alternative to FDSA is Simultaneous Perturbation Stochastic Approximation (SPSA) (Spall, 2005) which uses the same number of samples as RDSA while being more intuitive. In principle one could choose from a number of stochastic approximation algorithms available in the literature.

The detailed conditions for convergence of these algorithms differs but the basic constraints for the measured scalar field as stated in section 7.5.2 are the same for all algorithms from this family. Thus, the discussion related to wireless communication in section 7.5.2 is valid for the whole family of algorithms.

Then there are a lot of ad hoc gradient based algorithms, some of which are presented briefly in section 4.4, for which the discussion on the noise and the estimated errors in section 7.3 and section 7.4 respectively can be applied or are very similar, but for which no formal proofs of convergence exist. The discussion here can, however, be a good guideline for the design of such algorithms.

7.6. Experiments

In order to show how to overcome the problems of small scale fading the algorithm was implemented using a ground based robot. A ground-based robot was chosen because the

[2]These were not known to the author at the time when first using the FDSA framework to show convergence for this family of algorithms.

Figure 7.4.: RobuLAB 10 with plastic rod for antenna placement

effects of small scale fading are much more severe for a ground-based robot than for a flying robot since it can position itself with a much higher precision (also see the discussion in section 7.3.3).

The algorithm was first implemented with four measurements for every gradient estimation, following its original formulation. The resulting behavior was random-walk like and didn't show any convergence. The most likely reason for this behavior is small scale fading because of its local periodic nature and the resulting effect on this naive gradient estimation.

Since a robot has to move from one sampling point to the next while executing the algorithm, this movement can be exploited to allow measurements continuously while moving along the two axes of the gradient estimation. The finite difference estimation of the derivatives was supplemented with a linear model of the data collected along these lines, which basically uses the same local linearity assumption as finite differences. The model is fitted to the data collected while moving and used as an estimate of the derivative. If the model is fitted over an interval larger than the wavelength of the wireless communication, the impact of small scale fading is mitigated. A schematic example of this procedure is depicted in fig. 7.3.

A Robosoft RobuLAB 10, as depicted in fig. 7.4 carrying a laptop with an attached USB WLAN card was used as a mobile node and a second laptop with the same configuration as the network node. IEEE 802.11g was used as the communication standard and all antennas were positioned well above the ground and above the robot using plastic pipes. All calculations and the packet analysis were done in Python.

The trajectories of two indoor example runs of the experiment are depicted in fig. 7.5 and

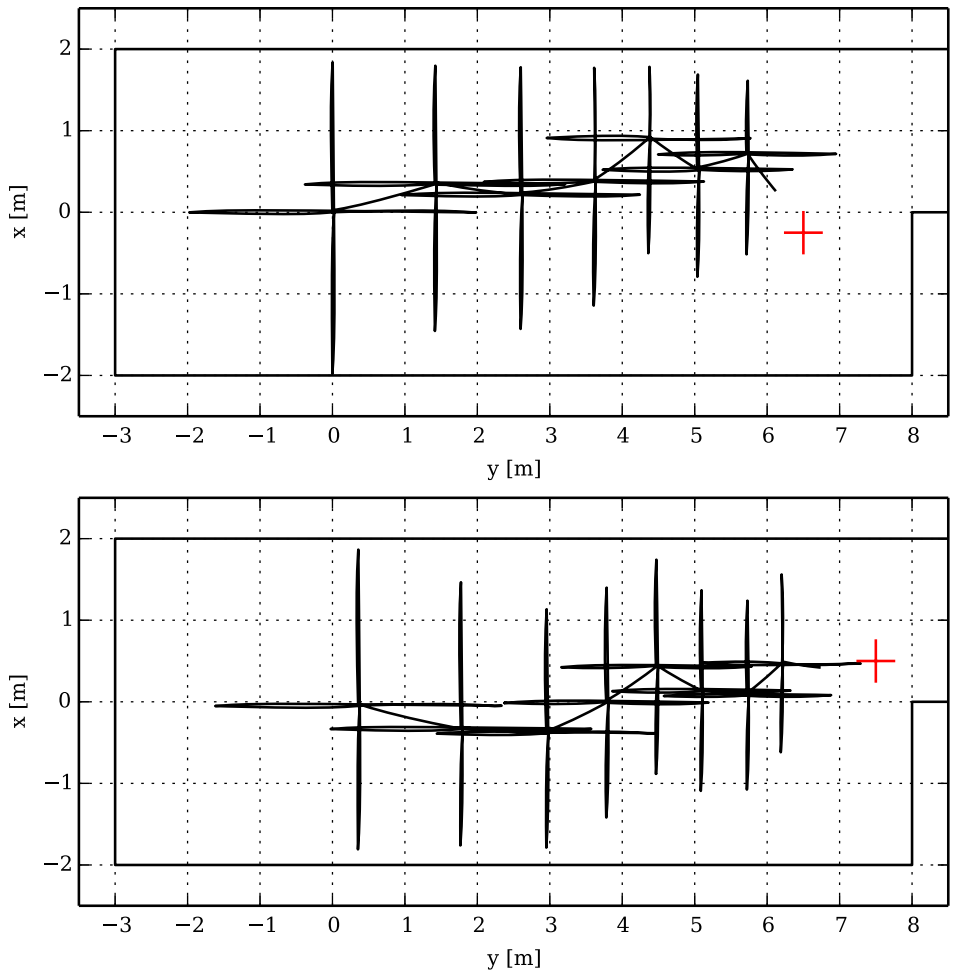


Figure 7.5.: Depicted are the complete trajectories of the robot in black, the position of the measured network node as a red cross and the physical boundaries of the room as a thin black line for two experiments. The trajectories of the estimates of the maximum are highlighted in red.

show good convergence of the position of the robot towards the network node. An indoor scenario was chosen because the effect of small scale fading is enhanced by a lot of scatterers like walls, furniture or metal doors

This experiment shows that the effects of small scale fading can be dealt with very well even in this worst case of a ground based robot in an indoor scenario with many multi-path effects.

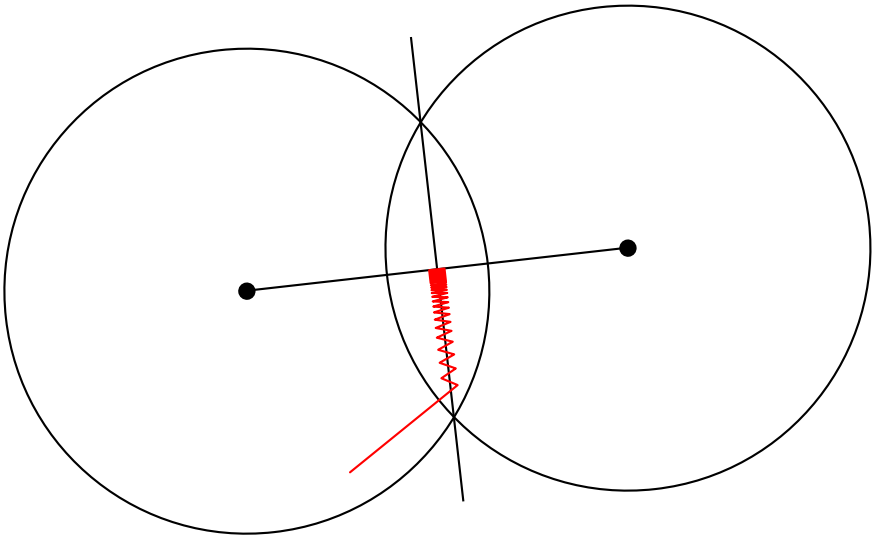


Figure 7.6.: Two network nodes are depicted as black circles. The two lines represent the two parts of the objective function, the points of maximal sum of both signal strengths and the points of equal signal strength, respectively. The hypothetical example run is marked as a red line.

7.7. Adapting the Algorithm to Complex Objectives

More complex objectives than finding a network node can be addressed using the proposed algorithm. For this, a target function, which satisfies the conditions stated in section 7.5.2 has to be constructed in terms of signal strength measurements only.

As an example objective, a typical task in network robotics, namely bridging two network nodes, was chosen, i.e., moving to the position between the nodes with maximum signal strength and equal signal strength to both nodes. This task often occurs when for example two separated networks have to be connected by bridging the gap between the two closest nodes of the two networks..

For this task the objective function is defined $g(\vec{x})$ as

$$g(\vec{x}) = \|f_1(\vec{x}) - f_2(\vec{x})\| - \|f_1(\vec{x}) + f_2(\vec{x})\| \tag{7.7.1}$$

which can be written (assuming $f(\vec{x}) >= 0$ which is true for signal strengths) as

$$g(\vec{x}) = \begin{cases} -2f_2(\vec{x}) & f_1(\vec{x}) > f_2(\vec{x}) \\ -2f_1(\vec{x}) & f_2(\vec{x}) > f_1(\vec{x}) \\ -\|f_1(\vec{x}) + f_2(\vec{x})\| & f_1(\vec{x}) = f_2(\vec{x}) \end{cases} \tag{7.7.2}$$

and only consists of signal strength measurements. This objective function has one global minimum at the desired point with properties as stated above. A hypothetical example run of this algorithm is depicted in fig. 7.6.

In principle, the behavior of this algorithm shows two stages. First the robot moves towards the line — in two dimensions, in three dimensions this is a plane — of equal signal strength to both nodes and then oscillates about this line towards the point of maximum signal strength. This can be seen as a general gradient descent in the first stage and then a line search — or a one dimensional gradient descent — in the reduced space of this line.

This objective function has the same properties as the signal strength itself, satisfying the constraints of the algorithm. This is true because noise is added before the absolute value is calculated. However, close to the area of equal signal strength of both nodes, both signal strengths cancel out in the first term of eq. (7.7.1). This results in the absolute value to be only calculated from noise, thus positively biasing it which in turn violates the constraint on the noise of having an expectation value of zero. Thus, in this area convergence cannot be guaranteed analytically. Nevertheless, this is not of any practical relevance because the problem reduces to a lower-dimensional gradient descent algorithm in this area.

This idea of adapting the gradient formulation to more complex objective by re-defining the objective function will be taken up again in chapter 8 in a slightly modified form to suit the different base algorithm. Specifically in section 8.5, real experiments are conducted and show the feasability of this approach.

7.8. Conclusion

This chapter showed that network robotics can make use of algorithms based on stochastic approximation, working with signal strength measurements, for tasks such as navigation. Estimates of the precision and statistical properties of gradients calculated from signal strength measurements were established. Signal strength measurements as well as motor noise were physically motivated and its effects on the convergence of these algorithms were discussed in depth. The algorithm was also implemented experimentally to show how to deal with some of the specific challenges posed by network robotics. Additionally, it was illustrated how more complex objectives can be formulated in this framework.

This work was presented in (Blum and Hafner, 2015)[1].

Chapter 8

Active Exploration of Wireless Sensor Networks

8.1. Introduction

Traditional algorithms for robots who need to integrate into a wireless network often focus on one specific task. Here, a simple, adaptive and reusable algorithm is presented for real world applications in this scenario. Starting with the most basic function for mobile wireless network nodes, finding the position of another node, an algorithm able to solve this task is introduced. Then it is shown how this algorithm can readily be employed to solve a large number of other related tasks such as finding the optimal position to bridge two static network nodes. To achieve these tasks, the algorithm has to be able to deal with the very noisy nature and the other physical characteristics of wireless networks. For a more in-depth discussion of the specific challenges posed by wireless networks, refer to chapter 2, specifically to section 2.2.2 and section 2.3.

To solve a source seeking task without directional sensing, i.e., purely scalar measurements of, for example, signal strength, often gradient-based methods are employed. Gradient-based methods are based on the implicit Taylor expansion of the underlying signal strength function. Using the gradient means using a local linear approximation of this function, which is fitted every iteration to the current local data. This is usually done directly on noisy data, and under some constraints, convergence for this class of algorithms can be proved (Atanasov et al., 2012; Blum and Hafner, 2014). One precondition for convergence of these algorithms is that local extrema, which can result from small scale fading, have to be mitigated else the algorithms can get stuck in these extrema. This can be achieved using multiple samples over an area larger than the wavelength of the signal, for example, by fitting a plane to the data instead of directly calculating gradients using finite differences.

To improve on this method, instead of an implicit model as assumed by a local linear approximation as done by gradient-based methods, a more elaborate kind of explicit model can be employed. By making the choice of this (internal) model explicit and using all the available data, local maxima can be directly detected and their effects mitigated. This also means that the shortest possible path to the maximum can be chosen directly instead of following the gradient. The price for this additional information is increased computational

[1]For a detailed breakdown of the contributions of the different authors refer to section 1.5.

and memory cost which is why a good strategy is to update the model only when necessary and not every iteration as for a gradient-based method. This is also due to being able to explicitly check model predictions against real measured data, leading to a measure of how good the model predictions are, i.e., to a measure of when the model should be updated. Additionally, using an ϵ-greedy strategy guarantees that the robot cannot get stuck in a wrongly perceived local maximum, which might be mistaken as a global maximum by the internal model. This strategy is a standard strategy employed in the field of reinforcement learning to balance exploration against exploitation (Sutton and Barto, 1998).

In general, the concept of internal models originates from control theory but has been introduced in the fields of biology Wolpert et al. (2011); Haruno et al. (1999) as well as in robotics Demiris and Khadhouri (2006); Schillaci et al. (2012a). For further in-depth discussion on internal models refer to chapter 3. Internal models are often used usually consist of a pair of forward model (predictor), which predicts sensory states as a consequence of motor commands performed at a current state, and inverse model (controller), which provides motor commands leading to a desired sensory state. In the context of this algorithm, forward models, which will be called internal models for the sake of simplicity, are the most interesting ones. The inverse models in this work are predefined mappings of positional information, signal strength and steering commands as defined in section 8.2. A prominent hypothesis in Cognitive Robotics and Neuroscience states that situations beyond a certain minimal complexity can only be effectively handled by utilising agent-internal models (internal simulations) Little and Sommer (2013). Internal simulations can cope with noise and delay that is inherent in most biological and robotics systems.

Internal models can furthermore be reused since they encode knowledge of the agent. Internal models of the signal strength distribution of two nodes learned while trying to locate both nodes could, for example, be reused in the task of trying to optimally bridge these two nodes. The concept of internal models is also agnostic to the actual representation of the internal model, so representations of varying complexity and statistical power can be used depending on the available resources and prior knowledge. The representations of internal models reach from simple k-Nearest Neighbors over linear models or MLP to Gaussian Process models. Some of these models are discussed briefly in section 3.3 and section 8.3.2.

In this chapter, first a meta-algorithm inspired by autonomous robot learning strategies using the concept of internal models is introduced in section 8.2, which yields a class of source seeking algorithms for mobile nodes. The implementation of these algorithms is then presented in section 8.3. Using this implementation, the effectiveness of this algorithm is demonstrated in real world experiments in section 8.4 using a physical mobile robot and standard IEEE 802.11 wireless LAN in an office environment. Then it is illustrated in section 8.5 how more complex tasks, which might be encountered by mobile wireless nodes, can be encoded in the same framework and how the introduced algorithm can solve them. These tasks can be direct (cross layer) optimization tasks or can also encode more complex tasks such as bridging two network nodes. The bridging scenario is chosen as an example, implemented on a real physical robot, and it is shown how the robot can solve it in a real world experiment.

8.2. Active Exploration Algorithm

First, a high-level view of the (meta-)algorithm is presented. This algorithm was designed with the task of finding the source of a wireless signal in mind but can readily be extended to a host of different tasks as will be shown in section 8.5.

Algorithm 8.2.1 Main Algorithm

Require: x,y,r as asynchronous messages
  do random movement
  while not at source **do**
    parse messages
    if abs(mean RSSI over last second - model prediction at current position) $>$ error threshold **then**
      discard current model
      learn new model on all available data
    if random() $> \epsilon$ **then**
      go with min(dist to goal, step width) towards global minimum of the model
    else
      do random movement

The algorithm is presented schematically on a high level of abstraction in algorithm 8.2.1. While executing the algorithm, the robot receives a constant and asynchronous stream of messages from its various sensors, of which the current 2D position and the signal strengths of measured packets are used by the algorithm. At the beginning of each iteration of the algorithm, these messages are parsed and the positions interpolated to fit the timestamps of the signal strength measurements. The robot is capable of localizing and navigating autonomously, which also means that movements are restricted to legal movements, i.e., ones which are not part of a wall or outside of the known map. For details of the autonomous navigation refer to section 8.3.1.

The robot starts the algorithm with a random movement, which corresponds to a flat prior on all possible options. The initial movement is restricted to twice the regular step width and facilitates the initial learning of the internal signal strength model (see section 8.3.2). A constant step width of $1\,\mathrm{m}$ was chosen for all experiments.

At the beginning of each iteration, all messages in the buffer are parsed and added to the available dataset. Then the model prediction error at the current location is calculated and compared against the mean signal strength over the last second. This low-pass filtering is implemented to mitigate some of the measurement noise in order to make the algorithm more conservative. If the prediction error surpasses an error threshold (a threshold of $3\,\mathrm{dB}$ was used throughout), the current model is discarded and a new model is learned based on all available data. In principle the model could be discarded in every iteration and replaced with a new model based on the latest data, but in order to keep computational cost to a minimum, especially for more complex models, a model is retained as long as its performance does not degrade.

The next step implements what is known as the ϵ-greedy strategy in reinforcement learn-

ing (Sutton and Barto, 1998) and is one of the most simple strategies to balance exploitation and exploration of an agent. The strategy executes the action with the highest reward with a probability of $1 - \epsilon$ and a random action with a probability of ϵ. Additionally, ϵ is annealed from 1.0 to 0.1 during the first couple of iterations. ϵ is annealed like $\epsilon = 0.9e^{-\alpha n} + 0.1 \propto e^{-n}$ where n is the number of iterations and α such that $\epsilon = 0.5$ at $n = 5$.

If a greedy step is selected, a grid search over the current model for all legal positions is performed to locate the global maximum. Simple grid search was chosen because the prediction steps of internal models are usually fast enough in relation to movement speeds and model learning to allow for a brute force approach. Any standard optimization method could be used as a drop-in replacement. The robot then either moves directly towards the maximum if it is not further away than the step width, or one step width towards the maximum. If the step leads to an invalid position, a random search on the line between the current position and the maximum starting at one step width distance is performed. The first valid point on this line that the search finds is chosen, i.e., either directly in front or behind the obstacle blocking the step. This means that the probability of choosing a step in front of the obstacle in relation to the one to go behind the obstacle is controlled by how close the robot is to the obstacle in relation to the size of the obstacle.

An ϵ-step means a random movement. This random movement is implemented in a novelty-driven fashion by choosing only points which the robot has not been before. This is implemented by checking if a possible random movement is a minimum distance of around the size of the robot away from any points of the past trajectory of the robot and discarding those options. Points outside the known map are also discarded because of practical reasons. A point $\vec{x}$ is chosen randomly under these constraints from a probability distribution $P(\vec{x}) \propto e^{-\|\vec{x}-\vec{x}_0\|}$ where $\vec{x}_0$ is the current position of the robot.

As for all optimization problems, there is no canonic way to terminate the optimization algorithm. The most straightforward options would be a cut-off signal strength, which can be problematic because of noise, a maximum number of iterations or manual stopping, which was used in the experimental examples. The algorithm itself will stop moving the robot once a global maximum of the model has been reached. The algorithm continues and will occasionally lead to ϵ-steps which drive the robot away from the maximum. If the predicted maximum is truly a maximum, the algorithm will lead the robot back to this maximum after executing the ϵ-step. If however the new measurements collected by this ϵ-step contradict the model predictions, the model will be updated and possibly predict a different maximum. Thus global convergence can be assumed, which also justifies the manual stopping of the algorithm once the robot has reached the known global maximum.

In general the parameters of the algorithm are not critically important for convergence, they mainly balance exploitation against exploration and are thus to some degree purely design choices. Different parameter configurations were tested without changing the behavior of the algorithm fundamentally. A configuration which worked well with the constraints for computational speed, speed of the robot and the test environment was chosen empirically for the experiments.

Figure 8.1.: TurtleBot 2 with USB WLAN adapter

8.3. Implementation

8.3.1. Hardware and ROS

A TurtleBot 2[2] was used as a mobile base. The algorithm is implemented as a ROS node[3] using a ROS node for capturing and analyzing the packets [4]. The ROS navigation stack[5] is used for navigation, localization and path planning using the Microsoft Kinect and odometry. The navigation is facilitated by a pre-learned map using OpenSlam's Gmapping[6] and amcl[7]. As a communication medium, off-the-shelf usb `IEEE 802.11` WLAN dongles were chosen and tcpdump/libpcap[8] are used for capturing packets. For all captured packets sent by the correct MAC-address, RSSI values are extracted from the Radiotap headers.

8.3.2. Internal Models

In order to limit the computational cost of learning an internal model, the maximum number of samples was restricted to 10.000, which were drawn randomly from the complete dataset. In the experiments $30,000$ collected samples was easily exceeded. As instantiations of the internal models, ridge regression and MLPs were used as two different machine learning

[2] `http://turtlebot.com/`
[3] `http://github.com/azz2k/wifi_im`
[4] `http://github.com/azz2k/wifi_sensor`
[5] `http://wiki.ros.org/navigation`
[6] `http://wiki.ros.org/gmapping`
[7] `http://wiki.ros.org/amcl`
[8] `http://www.tcpdump.org/`

techniques.

Local Ridge Regression

Ridge regression is a linear least squares model with a Tikhonov regularization (Tikhonov, 1943). A local version of this is employed by only fitting it to data inside inside some radius to have a similar model as used in gradient-based algorithms. Ridge regression as implemented by scikit-learn (Pedregosa et al., 2011) was chosen with a local radius of $5\,\mathrm{m}$ and regularization parameter $\alpha = 1.0$. No preprocessing of the data was used.

Even though using this model is similar to the idea of a locally linear Taylor approximation as used in gradient-based methods, it is superior in that it averages over an area, mitigating the effects of small scale fading and noise and in that it is regularized in order to prevent instabilities and overfitting.

Multilayer Perceptron

As a second example of an internal model a MLP with two input neurons, a hidden layer of 100 sigmoid neurons and one linear output neuron was chosen. All layers also have access to a bias node. Although this might seem like an overly large network, it is a requirement to deal with the complex dynamics of wireless communication in an office environment due to of the complex physical configuration (walls, furniture, etc.). Classic backpropagation with a small momentum term[9] for three epochs was used for learning. A fixed number of epochs was chosen to limit computational cost. Even though checking for full convergence and cross-validation are best practices in machine learning, those were omitted since the algorithm checks the predictions of the model against real measured data constantly and relearns the model if the prediction errors surpass a threshold. The network and learning was implemented using PyBrain (Schaul et al., 2010). The data was pre-processed to remove its mean and scale it to unit variance because input variables of with these characteristics are assumed by this implementation of MLPs.

8.4. Experiments

All experiments were performed in an ordinary office environment with about 20 different active access points and countless clients in the $2.4\,\mathrm{GHz}$ band in the vicinity of the test environment. The source node was emitting around 200 dummy packets per second which were then captured by the robot.

A complete mapping of the experimental area in terms of RSSI using the discussed setup was also performed. The resulting hexagonal histogram is depicted in fig. 8.2. The bins of the histogram are larger than the wavelength of the wireless signal which means that it represents pure path loss and no interference effects such as small scale fading because the bins effectively perform a low pass filter on a scale larger than the wavelength. A average logarithmic Gaussian noise with a standard deviation of $4\,\mathrm{dB}$ was measured in accordance to theory.

[9] learning rate 0.01 and momentum 0.1

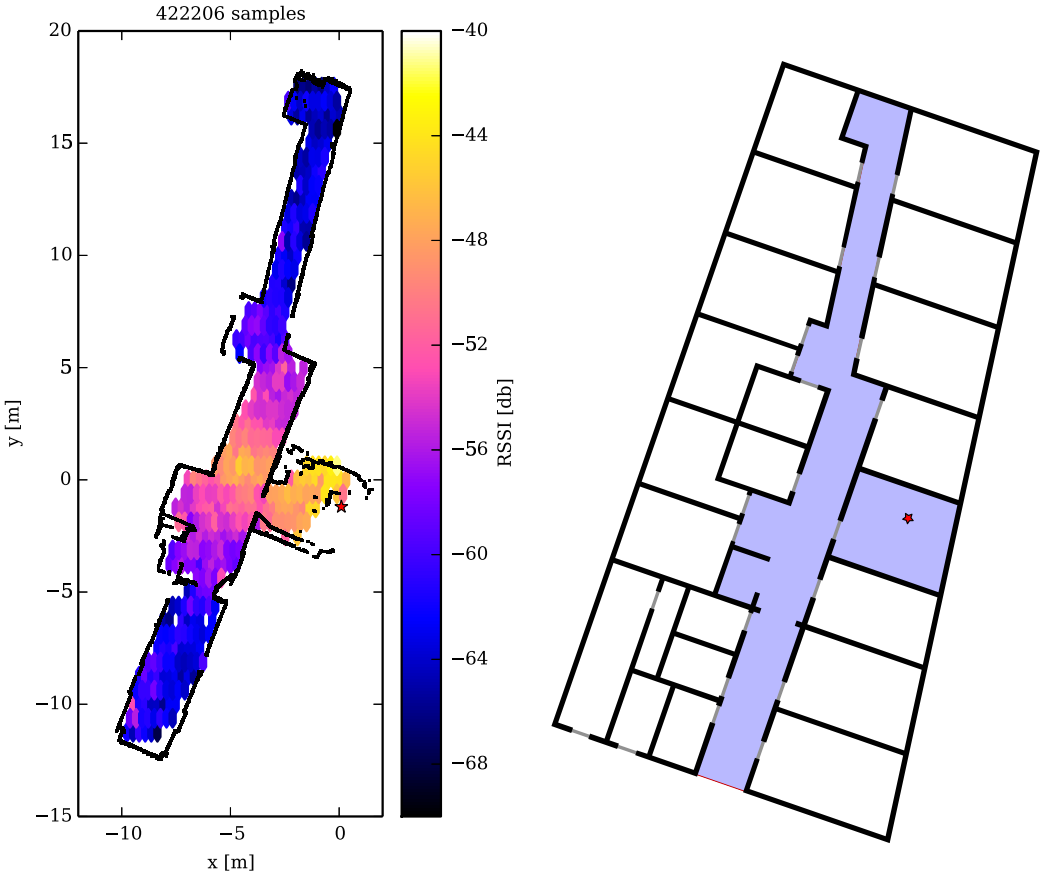


Figure 8.2.: Figure 8.2a shows the histogram of 422206 RSSI measurement samples of target function f collected with a mobile robot in an office environment. The position of the static source node is depicted as a red star and walls are depicted as black lines. Figure 8.2b depicts the corresponding floor plan of the office environment in which the experiment was conducted. The area covered by the measurements is shaded in light blue and the position of the static source node is depicted as a red star.

To accurately model small scale fading and other interference effects, a very accurate simulation or even the complete solution of Maxwell's equations is necessary. This is computationally costly and requires accurate knowledge of material properties of walls etc. (see the discussion in section 2.1.3). Effective models on the other hand are cheaper to calculate but only represent an abstract, stochastic model of the environment. Thus, working directly with real world experiments is to be preferred in order not to lose or miss systematic physical effects such as small scale fading; following the quote by Arturo Rosenblueth and Norbert Wiener „The best material model of a cat is another, or preferably the same, cat."(Rosenblueth and Wiener, 1945, p. 320).

Since the experiments were performed during regular office hours, a lot of disturbances such as people walking, opening and closing doors, changing network load etc. affected the measurements and path planning. Additionally, the algorithm itself is stochastic. Therefore, a large number of experiments would have to be conducted to gain statistically sound results. Some typical experiments are shown instead for practical reasons.

Five typical runs for the two different internal model implementations as discussed in section 8.3.2 are depicted in fig. 8.3. For all experiments the robot starts at the same initial position. The plots show trajectories, points where the model was updated, as well as the signal strength measurements as a function of the way traveled. Duration times are not shown since they vary wildly because of the robot stopping for people, different training times, etc.

The trajectories show several distinct variations which can shed some light on how the algorithm works. In the beginning of each experiment the robot does not possess any knowledge about the signal strength distribution and starts with an initial random step. For the next five iterations ϵ is higher than 0.5 and since the ϵ-steps are novelty driven, the probability of continuing in the same direction as the initial random step is higher than the probability of turning around. Once ϵ is approaching its final value of 0.1 and the robot has collected enough samples of the signal strength distribution, it either turns around if it initially went into the wrong direction or it continues on towards the maximum. Once the robot passes the room where the target node is located, it shows a similar behavior. Either the internal model correctly predicts the position of the target node and the robot directly enters the room or the robot passes by the room but then turns around after the prediction error of the internal model measured against the collected samples surpasses the error threshold and the model is updated with new samples.

It is clear what a challenge the task is when considering the measurement history of the experiment. Especially far away from the source, the gradients are much smaller than the noise of the measurements. Using finite difference gradient methods would only work for impractically large measurement steps or would lead to very noisy trajectories close to biased random walks. In contrast, both internal models seem to be able to cope with the noise and possible local maxima.

The particular choice of the internal model does not seem to make much of a difference though. This may be due to the simple (and convex) nature of the underlying function, which is essentially the path loss function measured and depicted in fig. 8.2. The choice of a concrete model might be more important when working with more complex target functions as discussed in section 8.5. In addition to the two exemplary models shown here, also Sup-

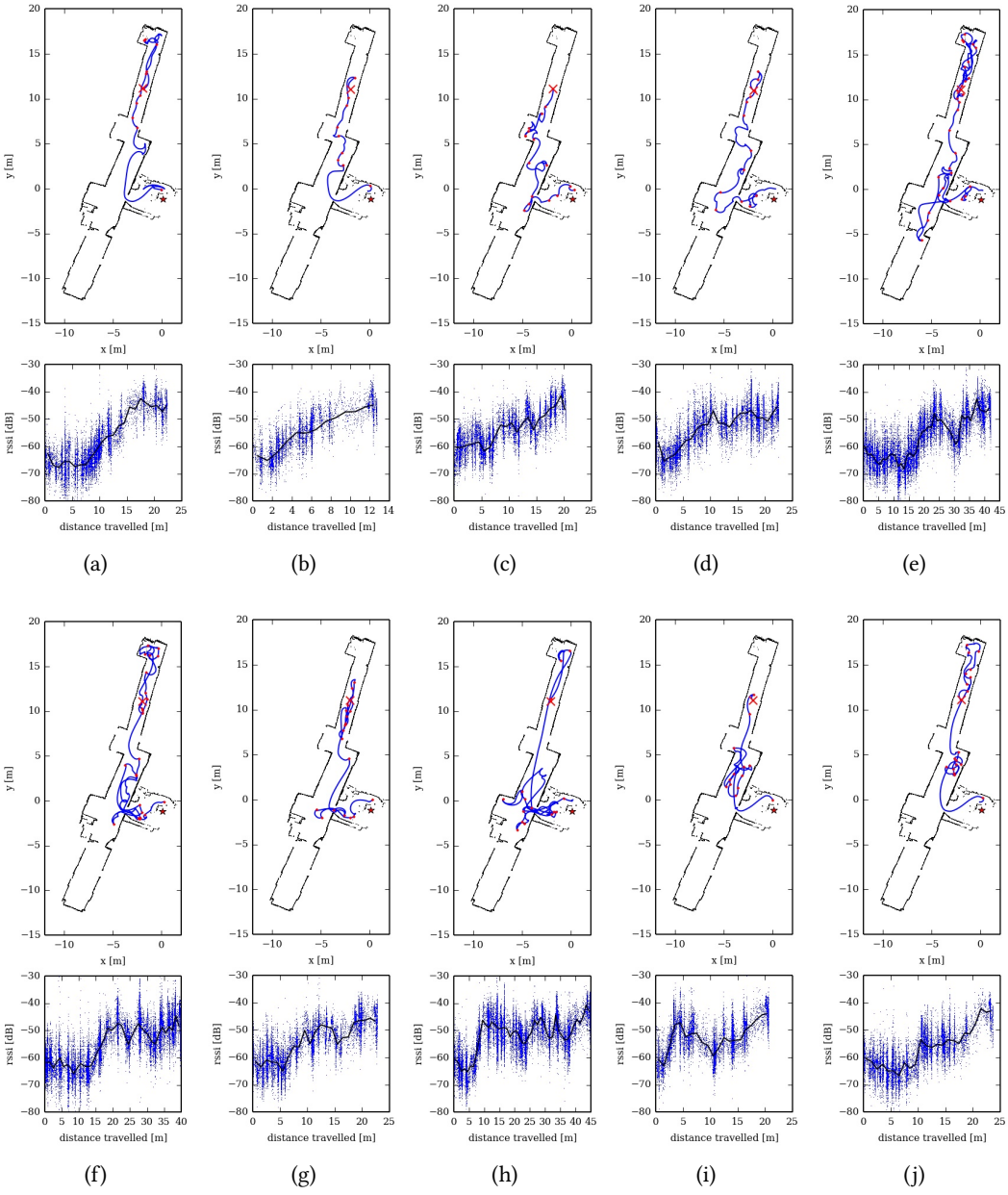


Figure 8.3.: Figures 8.3a to 8.3e depict experiments done with Ridge regression as the internal model, and figs. 8.3f to 8.3j depict experiments done withe an MLP as the internal model. Top panels shows trajectories on the map of the used office space and lower panels show the history of the measured RSSI values as a function of the distance covered and the moving average over 1 m. The red star denotes the position of the source. Red circles on the trajectory denote model updates. The initial position of the robot is marked with a red cross.

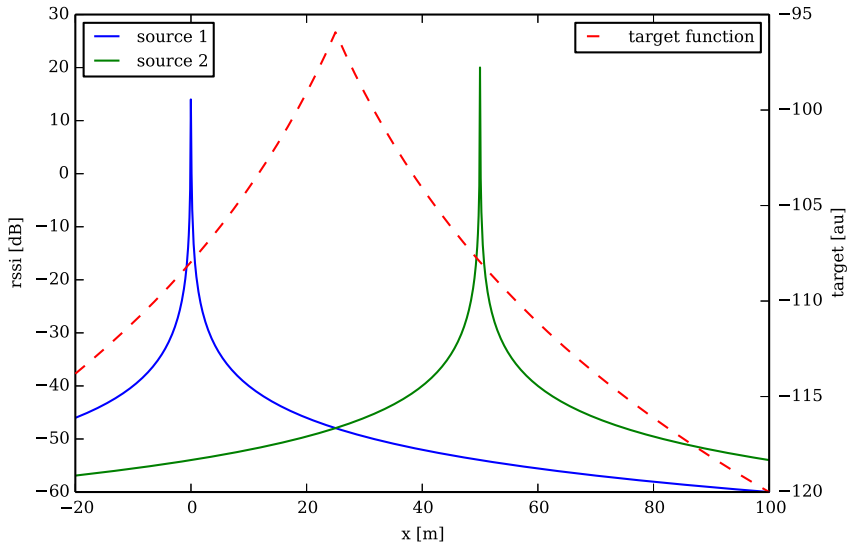


Figure 8.4.: Example target function g for finding the optimal position to bridge two network nodes as a 1-dimensional simplification. The signal strength of the two nodes with a toy path loss models are plotted on the left y-axis. The target function g is plotted on the right y-axis.

port Vector Regression, Kernel Ridge Regression as well as different other configurations of an MLP, k-NN and Radius Nearest Neighbor Regression were tested. All representations converged albeit with different convergence speeds. The choice of the meta-algorithm, i.e., constant model validation and novelty driven ϵ-greedy search, will likely lead to convergence of the algorithm regardless of the actual model representation as long as the particular model is in principle able to represent the target function.

8.5. Extension to Complex Goals

8.5.1. Idea und Theory

The presented algorithm can readily be extended to solve different tasks by noting that in essence it is just maximizing an unknown function with access to scalar measurements. The most straightforward extensions in the networking context would thus be to replace signal strength with any other interesting metric like for example PDR or Bit-Error Rate (BER). These metrics might be more noisy and contain a lot of local minima but the algorithm is able to cope with that by design.

More complex tasks can be easily constructed using compound metrics. As an example the task of bridging two network nodes is considered. Practically, this means moving to the point with the maximal but equal signal strength to both nodes. A single metric fulfilling this criteria can be constructed using the signal strengths of both nodes using a target function

g like

$$g = -|\mathrm{RSSI}_1 - \mathrm{RSSI}_2| - |\mathrm{RSSI}_1 + \mathrm{RSSI}_2|$$

where the indices 1 and 2 correspond to the two nodes to be bridged. This target function g with toy signal strength path loss functions for both nodes is depicted in fig. 8.4.

The target function shows a clear maximum at exactly the center between the two source positions, so maximizing this target function leads to maximal and equal signal strengths to both nodes.

This is only a toy example and other task-specific metrics can be designed in a similar manner making this algorithm very versatile in formulating and solving tasks for mobile nodes in wireless networks. A cross-layer approach incorporating measured quantities from different OSI levels would be especially interesting. For more complicated target functions, which are by definition also more noisy because of the propagation of errors, more complex internal models are likely beneficial.

8.5.2. Exemplary Experiment

Exemplary experiments were performed with two network nodes and the target function g as shown above. One of the nodes was the one used for the earlier experiments and the second one was placed in a different office. Both nodes were simple USB wireless adapters sending packets. Again the experiments were conducted during regular office hours. Figure 8.5 shows the histogram of the measured values over all experiments conducted. As expected the function has one global maximum between both nodes.

Using the target function g means that the noise of the measurements of both sources are combined following the law of propagation of error, which yields for the case of noise of the same amplitude but independent for both nodes and the use of the target function g twice the noise of a single node, which is around $4\,\mathrm{dB}$. A standard deviation of about $9\,\mathrm{arb.unit}$ was measured for the experiments which agrees well with the predicted error.

This also indicates that the magnitude of measured gradients was be less than the noise of the experiment, which means that a gradient descent algorithm would show a behavior similar to a random walk with drift instead of a noisy gradient descent. Convergence would not be impacted by this high error but convergence speed would be low.

Figure 8.6 depicts an exemplary experiment. A MLP was used as the internal model with the same parameters as for the single source experiment but the error threshold was changed to $7\,\mathrm{arb.unit}$ to account for the increased noise. The trajectory of the robot converges to the area of the maximum of fig. 8.5. The experiment was terminated manually as discussed in section 8.2. The large noise levels in combination with the ϵ-steps lead to rather unstable convergence at the maximum. The magnitude of the noise can be seen in the measurement plot of fig. 8.6.

The behavior of the algorithm can certainly be improved by tuning parameters but the general idea of testing the internal model predictions against real measurements and refining it when necessary in combination with the ϵ-greedy strategy ensures convergence even without optimized parameters and in spite of very noisy measurements.

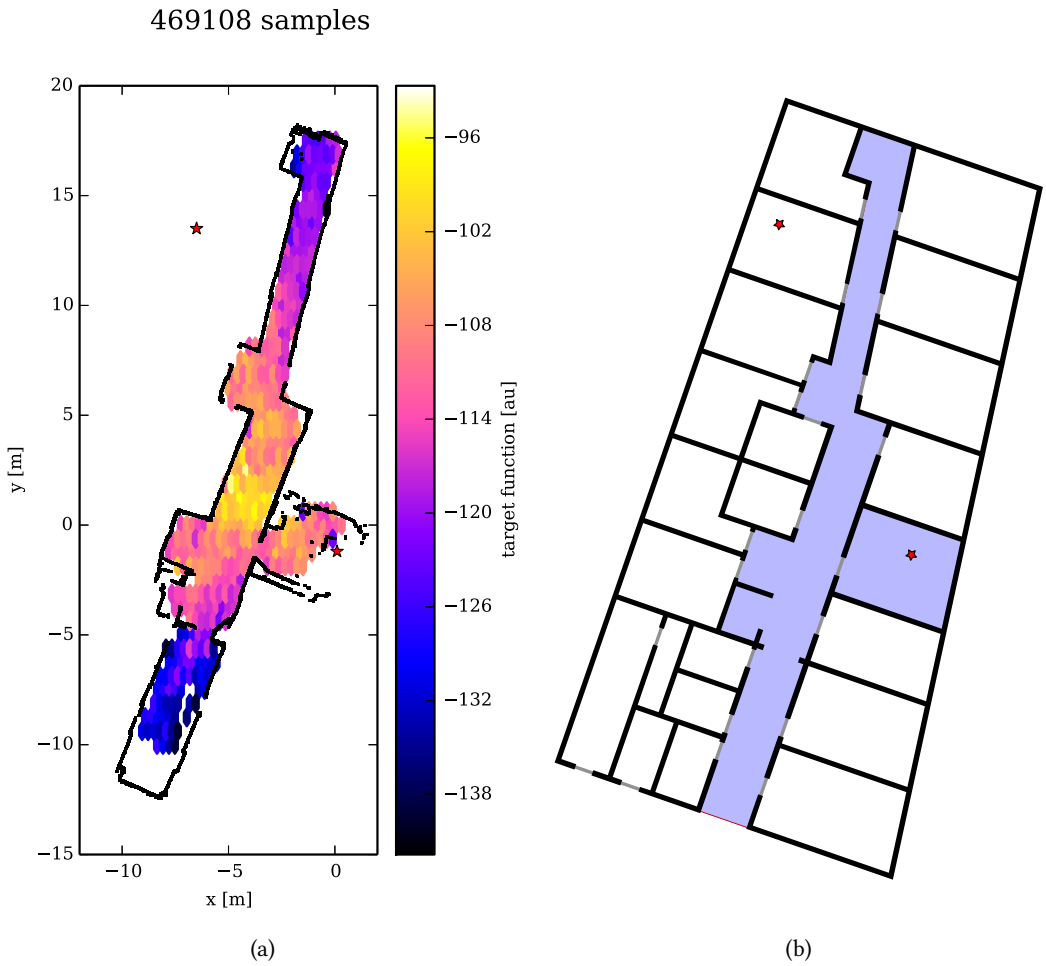


Figure 8.5.: Figure 8.5a shows the histogram of 469108 RSSI measurement samples of target function g collected with a mobile robot in an office environment. The position of the static source node is depicted as a red star and walls are depicted as black lines. Figure 8.5b depicts the corresponding floor plan of the office environment in which the experiment was conducted. The area covered by the measurements is shaded in light blue and the positions of the two nodes to be bridged are depicted as red stars

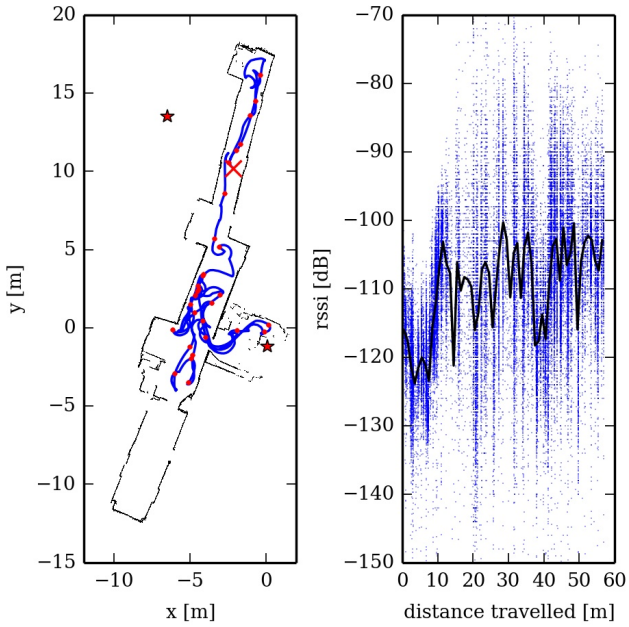


Figure 8.6.: The left panel depicts the trajectory of the robot for one experiment using the target function g. The initial position of the robot is marked with a red cross. The positions of the two nodes which have to be bridged are depicted as red stars. Model updated are denoted by red points. The right panel shows the measurements of the target function as a function of traveled distance and the moving average over $1\,\mathrm{m}$ of the measurements.

8.6. Conclusion

This chapter introduced a class of algorithms to solve a number of tasks related to network robotics, specifically to autonomous mobile network nodes interacting with a wireless network. Beginning with the most basic task of source seeking, the general algorithm was shown to be effective in real world scenarios by a series of experiments with a real physical robot in an office environment. It was then shown how this algorithm can be extended to various other tasks, which can be encountered by a mobile network node, such as maximizing the PDR of a connection to a certain target node or bridging two network nodes. For the bridging scenario an exemplary experiment was conducted successfully.

Part III

Self-Organization of Multiple Robots

Chapter 9

Introduction and Related Work

9.1. Introduction

In the context of network robotics, mobile nodes are especially promising in the context of Wireless Sensor Networks (WSNs) as discussed in chapter 4, where they can be used in several different roles. During deployment of a WSN, mobile nodes can be used to optimize network parameters such as coverage or link redundancy or they can undertake the complete deployment process if all nodes are mobile. A WSN consisting purely of mobile nodes able to deploy itself in an autonomous self-organized fashion is especially attractive for disaster scenarios where such a WSN consisting of mobile nodes might be used as a quickly deployable emergency communication backbone if infrastructure is not available or destroyed. Furthermore, this optimization process does not have to end after deployment but can continue during operation. This also means that the WSN can be dynamically optimized depending on for example current demands on the network. Finally, mobile nodes can be used to repair a WSN in the event of node failures. This capability is essential for the case of spatially correlated node failures, for example caused by a fire, where several mobile nodes could be used to reconnect a partitioned network.

Typical WSNs consist of a large number of nodes, often several hundreds or even thousands. In the context of the Internet of Things, embedded devices and smart objects are expected to be connected in networks of potentially orders of magnitudes larger in size (Mattern and Floerkemeier, 2010). This means that most centralized algorithms have problems scaling to these sizes because of computational and communication constraints. Gupta and Kumar (2000) show that the communication capacity of a WSN decreases with node density[1] and show that in order to solve that issue, a node should only communicate with nearby nodes. Incidentally this is also one of the main features of algorithms for self-organization, for example swarming algorithms based on emergent behavior through loose coupling of neighboring nodes. For a more in-depth discussion on general ideas of swarm robotics, refer to section 9.2.

Furthermore, since such self-organizing systems consist of a large number of similar units,

[1] In this model, n nodes with fixed communication range and capable of transmitting W bits per second are optimally placed in a disk of unit area. Then each node can obtain a throughput to a randomly chosen destination, which scales like $\mathcal{O}\left(\frac{W}{\sqrt{n}}\right)$.

they are potentially more robust against failures of single units. This topic will be discussed and critically reviewed in detail in section 9.5.

These properties of swarm robotic approaches in WSNs will turn up again in the chronological literature review on this topic in section 9.3. A clear trend away from centralized algorithms towards distributed, self-organized ones has been going on in the last decade underlining the beneficial properties of these kinds of algorithms.

In the rest of this chapter, an overview of algorithms for self-organization with a focus on mobile nodes in a wireless network is given. Furthermore, a short review on existing real testbeds using mobile nodes is given to gain an understanding of the scale of real implementations. To the best of my knowledge, there are no real world applications of mobile WSN nodes besides the ones for research. One of the presumably main issues why such real world applications are scarce is then discussed and after a short review of possible solutions, one of the most promising ones is presented.

9.2. General Swarm Robotics

There is no general definition of what swarm robotics is. As only a rather specific aspect of swarm robotics is considered here, swarm robotics will in the following be simply defined as "groups of robots operating in a collective way without any centralized control". For more detailed discussions on the topic and definitions, refer to the work by Şahin (2005) and Beni (2005).

Algorithms specifically concerned with swarm robotics in wireless networks will be discussed in section 9.3 so only a very brief general introduction into swarm robotics, mainly consisting of references to the literature, is given here.

A good introduction into the topic combining biological examples and technical applications is discussed in the book „Swarm Intelligence: From Natural to Artificial Systems"by Bonabeau et al. (1999), which focuses on models of social insects and their applications. Even though more modern literature also often describes swarming in vertebrates like flocks of birds or shoals of fish, it gives a very good introduction into the topic of distributed designs

A more recent and comprehensive overview is given by Brambilla et al. (2013). They analyze the current state of swarm robotics from an engineering perspective in regards to methods and collective behaviors. Interestingly, they also note that there are almost no swarm robotic systems used in the real world and then look into possible reasons. They briefly mention that traditionally swarm systems are seen as potentially more robust than conventional systems but immediately add that this might only be true to a given extent. This issue will be further discussed in section 9.5 in further detail.

9.3. Network Swarm

Countless algorithms for swarm robotics in wireless networks have been developed. Instead of listing all the different algorithms for self-organization in WSNs, an overview over the historical development of this area is given in terms of survey papers of the last decade.

These survey papers are presented in temporal order. The method of using virtual physics will be mentioned several times. For a brief discussion refer to section 9.3.1.

Mills (2007) gives a good definition and review in the context of networks as to what self-organization means. He presents a short overview over possible design strategies and shows areas in which self-organization can be employed in wireless networks. The most interesting in the context of this work are structure formation and maintenance, sensor placement and resilience. He also raises the interesting question of phase transitions in WSNs (Krishnamachari et al., 2003) (for more discussions on the topic of phase transitions in swarm systems see also (Vicsek et al., 1995) and (Buhl et al., 2006)).

Younis and Akkaya (2008) discuss node placement strategies for WSNs to optimize the network for given requirements. They describe in brief two strategies for node placement. There are strategies performing this optimization at the time of deployment or for ongoing optimization while operation of the WSN. A large number of algorithms for different objectives and deployment schemes and so on are reviewed. Especially interesting in the context of this work is the second case of continuous optimization during operation of the WSN. Most of these algorithms are non-collaborative in the sense of explicit collaboration but loosely couple the different robots by interaction via their behaviors. Furthermore, they identify the scaling problem and find that methods solving this problem often lead to local, approximative solutions which then in turn often lead to emergent behavior. They then identify multi-node relocation as one of the biggest open research questions.

Wang et al. (2009) emphasize the importance of mobility for WSNs. Besides optimizing coverage, mobile nodes are also able to heal coverage holes as well as adapting the network to new tasks. Mainly two classes of algorithms are discussed: geometric ones based on Voronoi-diagrams or fixed location patterns and virtual physics based ones.

Wu et al. (2011) present cyber physical systems as a promising research direction. They define them according to „cyber physical systems bridge the cyber world (e.g., information, communication and intelligence) to the physical world through lots of sensors and actuators.“ Thus, these systems are inherently mobile and interact with (mobile) sensing networks. The also specifically review coverage and deployment issues in WSNs and mention mobile nodes, i.e., robots as one of the main solutions to these issues. They also give examples for concrete cyber physical system applications in different areas, for example intelligent transportation systems.

Zhu et al. (2012) survey strategies for solving coverage and connectivity issues in WSNs. Most interesting here is the topic of coverage deployment strategies, specifically dynamic coverage. The discussed algorithms are again mainly built on either virtual physics or graph-based. Additional to standard coverage problems, repair policies for coverage holes in a WSN are discussed. Such coverage holes can be created by spatially correlated failures of nodes, such as those caused by fires. This research area of coverage hole detection and repair is also highlighted as one of the key areas for future research challenges.

Younis et al. (2014) survey algorithms to tolerate node failures in WSNs. Note that this survey only deals with complete node failures and not with the more difficult case of partial node failures (see section 9.5). Nevertheless, the strategies to deal with those failures are interesting in the context of this work since most of them involve node mobility and the most interesting strategies are reactive, i.e., dynamic. For these strategies, distributed algorithms

are favored over centralized ones, because centralized ones would need an alternate way of communication with the WSN nodes besides the WSN itself, which might be, for example, partitioned by the failures, i.e., possibly not operable. A number of algorithms for single failing nodes are reviewed. For the case of multiple node failures, the case of spatially collocated node failures (such as in the case of a fire) is especially interesting because it needs either the introduction of a number of new nodes into the network or a global repositioning of the remaining nodes of the WSN to fill the gap, which lends itself to a distributed implementation. Common algorithmic choices are virtual physics and/or graph-based. Another interesting approach uses additional sensors, for example a camera to gain additional information to improve the algorithms. As an important future research questions, the authors list amongst others the ability to detect and/or quantify the scope of the failures in a distributed manner as well as security issues like trust between the nodes. Furthermore, the issue of real world evaluations is pointed out since until now almost all research is done in simulation[2].

From this retrospective of survey papers of the last decade, some trends can be extracted. Mobility is becoming more and more important in the area of WSNs. Algorithms are becoming more swarm-like and are based on emergent behavior to realize the beneficial properties of these kinds of algorithms such as scalability and so on. Focus has shifted from pure deployment of a WSN to ongoing optimization during operation and thus also to the capability to react, for example to complete node failures. This then often leads to distributed and/or localized algorithms because of superior scaling characteristics and resource efficiency. However, despite the huge body of past and ongoing research, very little real implementations exist, especially when it comes to mobile nodes. One probable reason for this is in my opinion reliability of real swarm systems as will be discussed in more detail in section 9.5.

9.3.1. Virtual Physics

The method of virtual physics, of which the most commonly used aspect are virtual potential functions (PFs), is often used in the context of WSNs for example for node deployment (Howard et al., 2002; Heo and Varshney, 2005). At the core of this method, virtual forces or PFs acting between the individual nodes, are designed in a way that all nodes behave collectively in the desired way. Since the forces can be designed in an analytical way, and because there are a lot of well studied examples in physics, this method is one of the most successful ones.

This idea of using virtual forces can be traced back at least as far as 1987 when Craig W. Reynolds showed an approach for simulating a natural flock of birds, which he calls Boids, for animation purposes at ACM SIGGRAPH '87 (Reynolds, 1987). This approach was based on virtual forces between the birds based on three simple rules for collision avoidance, velocity matching and flock centering respectively.

This method was systematized by Spears (2012) in his book „Physicomimetics: Physics-based swarm intelligence ". This book is also a very comprehensive review on the topic and contains a lot of design examples. It is a good starting point to design algorithms for swarm robotics achieving specific goals.

[2] I suspect this is due to the fact that the real world implementations lead to a lot even more difficult problems than what the algorithms are supposed to solve like for example partial failures

An interesting possible extension for the design paradigm of virtual physics, especially using virtual PFs, has been proposed by Edlund et al. (2011) in the context of complex systems research. They show a direct method to design isotropic PFs causing self-organization into arbitrary complex lattices like for example Kagome lattices. In the domain of swarm robotics such methods could be of special interest since the virtual PFs are not limited to physically plausible ones.

The method of virtual physics has also been combined successfully with other approaches such as for example by Wang et al. (2006), who presented a voronoi-based algorithm that uses a mix of a graph-based and virtual physics based approach for node deployment in a WSN.

Similar methods have been also used successfully to model animal swarms. For animal swarms, usually a set of biologically plausible rules are translated into a set of corresponding forces, which are then fitted to real measured trajectory data of the respective swarm. This has been successfully performed, for example for flocks of birds(Ballerini et al., 2008; Hildenbrandt et al., 2010) or fish (Hemelrijk and Kunz, 2005; Hemelrijk and Hildenbrandt, 2008). Also more complex effects can be modeled in a similar fashion as, for example, in the work of Buhl et al. (2006) on phase transitions in locust swarms or even to modeling escape panic in humans (Helbing et al., 2000). The very abstract underlying agent models used in these contexts are also known as self-propelled particles (SPPs) (Vicsek et al., 1995).

9.4. Real Experimental Testbeds

This section gives a brief overview on real testbeds for mobile WSN nodes. Mainly these testbeds can be separated into flying and ground-based mobile nodes but there are also some mixed ones. Underwater testbeds also exist but are excluded from this overview because of their specific communication issues due to the different medium (water or air). Furthermore, testbeds, which are only used for control theory experiments like coordinated acrobatic flying and do not specifically deal with WSN issues, are also excluded. Finally, this is not supposed to be a comprehensive survey but a sample to gain an understanding of the type, and especially the average size of real world WSN testbeds using mobile nodes.

A compact overview of the testbeds considered is given in table 9.1. Most of the testbeds are discussed in two recent surveys: Bekmezci et al. (2013) give a comprehensive survey on flying ad-hoc (sensor) networks and Jiménez-González et al. (2013) present a general survey on testbeds for ubiquitous robotics, which also includes a large number of WSN (related) testbeds.

Two basic features can be readily extracted from table 9.1: first, testbeds using flying nodes have only been developed in the last five years and second, the size of testbeds seems to be limited at around the order of magnitude of ten mobile nodes. For static WSNs, this number is at least an order of magnitude higher. Furthermore, general swarm robotics testbeds are often in the same order of magnitude in terms of number of robots. An exception are very simple robots such as presented by Rubenstein et al. (2012) and called Kilobot, which holds the current record in swarm size with 1000 robots (Rubenstein et al., 2014). These robots are however very limited in sensing, computational and movement capabilities and not suited

Project	Type	Reference	# of mobile nodes
SMAVNET	flying	(Hauert et al., 2010c)	10
SMAVNET-II	flying	(Rosati et al., 2013)	(3)
AUGNet	flying	(Brown et al., 2004; Dixon, 2010)	2
SUAAVE	flying	(Cameron et al., 2010)	(10)
sFly	flying	`http://www.sfly.org/`	3
sWarms			
MARS	ground-based	(Clark et al., 2003)	6
MVWT-I	ground-based	(Cremean et al., 2002)	6
MVWT-II	ground-based	(Jin et al., 2004)	12
MiNT-m	ground-based	(De et al., 2006)	12
MADNet	ground-based	(Reich et al., 2008)	9
(MIT CSAIL)	ground-based	(Correll et al., 2009)	9
COMET	ground-based	(Cruz et al., 2007)	10
Robomote	ground-based	(Dantu et al., 2005)	14
StAR	ground-based	(Obraczka et al., 2007)	4
Mobile Emulab	ground-based	(Fish et al., 2006)	6
ISRobotNet	ground-based	(Barbosa et al., 2009)	6
CONET	ground-based	(Jiménez-González et al., 2011)	6
MAGICC	mixed	(McLain and Beard, 2004)	10
MARS2020	mixed	(Chaimowicz et al., 2005)	8

Table 9.1.: Numbers in brackets either mean uncertain data (conflicting or vague numbers in the literature) and/or the testbed is not at the full stage of expansion yet. For mixed systems only the number of mobile nodes is listed. Also only testbeds which are concerned with WSN aspects are listed.

for testing algorithms developed for WSNs.

The factors limiting testbed sizes are certainly to some extent acquisition and maintenance costs, flying systems in particular require a lot of maintenance as failures often result in extensive repairs. However, in addition to these conventional reasons it can be shown that for a lot of types of algorithms, reliability of swarms decreases with swam size in the presence of partial failures, which then limits practical swarm sizes to be used with „naive “swarm algorithms. This fact will be discussed in detail in section 9.5.

9.5. Failures in Swarms

One of the most often cited features of swarm systems especially in swarm robotics is robustness due to the fact that a swarm solves a task using many independent individual units. Thus, according to the general consensus of the literature, failures of single swarm elements do not impact the overall operability of the swarm. A theoretical analysis of this situation as well as a case study in simulation and as real world experiments was performed by Winfield and Nembrini (2006). They show that in general swarm architectures are indeed robust to

complete failures of single robots but highlight that they are less tolerant to partially failed robots such as robots with failed motors with the rest of the robot working properly.

In continuation of this work, Bjerknes and Winfield (2013) present a case study using reliability modeling and complementary real experiments to show that for a swarm relying on emergent behavior, reliability drops quickly with increasing swarm size for worst-case failures, i.e., partially failed robots. Specifically for their case study, they show that the swarm becomes unreliable after a time much briefer than the mean time between failures of the individual robots.

In Addition to failing robots, malicious ones also have to be taken into account. (Higgins et al., 2009) investigate the security of swarm robotic systems. They give a comprehensive review of security challenges for swarm robotic systems, for example communication and identity management. Besides these conventional challenges, they also consider ones specific to swarm robotic systems, of which intrusion detection is especially interesting in the context of this work. Specifically, the example of one or more foreign robots infiltrating a swarm and maliciously affecting the desired emergent behavior is given. This is closely related to the effect of partially failed robots.

This effect has already been observed in natural swarms. A recent methodology in collective animal behavior research is to introduce robots into animal swarms (Faria et al., 2010). Using this methodology has shown that animal swarms can be manipulated: Halloy et al. (2007) could demonstrate how the collective choice of a shelter in groups of cockroaches could be influenced using robots introduced into the swarm. Four robots were able to manipulate a swarm of 12 cockroaches into choosing the worse of two offered shelters. Faria et al. (2010) showed that a single robotic fish was able to recruit and lead a swarm of 20 real fish. Since these early experiments a number of similar studies has been performed for different animal species, which have generated further evidence of the manipulability of collective animal behavior relying on emergence. One further interesting example is the study conducted by Abaid et al. (2013) who demonstrated that a single robotic fish was able to modulate the risk taking behavior of single real fish in the presence of a fake predator.

In order to cope with these challenges a way of recognizing and dealing with a failed (and/or malicious) swarm member is needed.

9.6. Safety and Reliability for Swarms

As discussed in the previous sections, real world applications of swarms are scarce not only because of the logistical and technical challenges but also because of their susceptibility towards partial failures as discussed in section 9.5. In order to make swarm robotic systems more usable, this aspect needs to be addressed. Some approaches have already been suggested in the literature and will be briefly discussed. In particular an approach of using internal models (see chapter 3) turns out to be especially interesting. It will be briefly introduced here and discussed in detail in chapter 10 and is the basis for the second part of this thesis.

9.6.1. Byzantine-Tolerant Approaches

Fault tolerant algorithms for the gathering problem with Byzantine fault models have been presented by Agmon and Peleg (2006) and Clement et al. (2012). Using very specific computational models for the robotic system, which describe the system from a distributed computing point of view, they rigorously show the fault tolerance of the proposed algorithms as well as its limitations. Bouzid et al. (2010) show fault-tolerance for similar models in one-dimensional space. There are more studies on gathering using different similar models and assumptions leading to different bounds for the number of failed robots in relation to the number of robots in the system (Bouzid et al., 2009).

Similar studies have been performed for different flocking algorithms. In the same computational model, as for most gathering algorithms, an algorithm for a flocking task can be shown to be fault-tolerant (Souissi et al., 2008). For a different computational model fault-tolerant shape-rotation flocking can be be shown to be fault-tolerant (Yang et al., 2009).

In general there are more studies with the similar approaches for problems such as consensus or synchronization which present algorithms tolerant to Byzantine fault models. However, those approaches are always bound to one particular algorithm and computational model and there is no generalized approach working for arbitrary algorithms and models.

Furthermore, in order to be able to make use of the tool sets from distributed computing, heavily abstracted models are necessary. For example, one of the most often used computational models is CORDA (Prencipe, 2001). In this model, robots are memoryless, asynchronous and homogeneous and perform a wait, look, compute, move cycle. There is no environment, no physics, and no communication between the robots, and sensing is performed by using the position of all other robots but not their identities. Sensor- as well as motor noise has been studied in very simple, i.e., bounded error, models (Cohen and Peleg, 2006). In general, the practical relevance of these models is therefore arguable.

Nevertheless, this approach is very interesting because the requirements for and the limits of these algorithms can be stated exactly and in closed form, which is not very common in the context of swarm robotic systems since most algorithms rely on emergent behavior. This might also lead to a deeper understanding of the reasons why some approaches are fundamentally not feasible. The algorithms could furthermore be used as a starting point for real world implementations and guide the design of more fault-tolerant algorithms. However, until now there has not yet been, to the best of my knowledge, any attempt at transferring the results of these studies to real world algorithms and systems.

9.6.2. Artificial Immune Systems

The area of Artificial Immune Systems (AISs) connects the disciplines of computer science, engineering and (theoretical) immunology. In essence AIS applies adaptive algorithms inspired by biological immune functions and results from theoretical immunology to problem solving in different areas. These areas are not restricted to ones directly related to immune systems such as fault detection. Dasgupta et al. (2011) give a survey on the historical development of the field and also review recent developments and methods.

AISs are well suited for application in swarm systems as „the immune system should be

viewed as an example of a swarm system " (Timmis et al., 2010, p. 248). As such they rely on local information and are distributed. The direct application to fault detection in swarm systems has been shown to be feasible.

In their work on fault detection in robot swarms, Lau et al. (2011a,b) use an algorithm based on a Receptor Density Algorithm to self-detect and classify failures in a foraging scenario. This algorithm can also detect and classify partial failures such as failed or deteriorated motors. The algorithm is inspired by T-cells signaling mechanism in the immune system and this algorithm was mapped to statistical classifiers for the failure classification. The mechanism behind this algorithm works by comparing the behavior of the robot to the one of other swarm members using local interactions.

This line of work has been extended to detect simultaneous failures of multiple robots in a swarm (Lau et al., 2013). As the original algorithm detected failures by comparing the behavior of a robot to its neighbors, the case of simultaneous failures in a swarm required some extension. This is solved by adopting a collective self-detection scheme for failure detection, which is based on comparing the behavior of a robot to a minimal number of neighbors. Failure classification is again performed by a Receptor Density Algorithm. The authors show that this approach can successfully extend the failure detection to multiple simultaneous failures in homogeneous swarms. They also show that the performance of the detection increases with swarm size.

These algorithms are purely data-driven and do not employ an internal model. Instead they use the behavior of neighboring robots, which in a homogeneous swarm should perform the same algorithm on the same embodiment, as a reference to cross-validate their own behavior. Therefore, these algorithms do not work in heterogeneous swarms or, for example, in human-robot interaction scenarios.

9.6.3. Control Theory

In the field of industrial control, execution monitoring — or fault fault detection and isolation (FDI) as it is called in the field of control theory — is well studied and used but not very common in robotics (Pettersson, 2005). Pettersson (2005) gives a survey on this topic and possible applications for autonomous mobile robotics. This topic is relevant insofar as the methods discussed are similar to the use of internal models for fault detection. The most commonly used approach of FDI in robotics is the observer-based one, which is basically a forward model used to generate expected system outputs from system inputs and earlier outputs, to be compared against actual system outputs. There are different approaches for that strategy depending on how the model is formulated and/or which variables can be measured and so on, and the survey gives a comprehensive overview of the different approaches. A very similar survey was performed by Qin et al. (2014) with a focus on swarm systems. They classify methods on the one hand by their topology, i.e., centralized, hierarchical, and decentralized and on the other hand on their methodology, i.e., qualitative and quantitative, which is then again sub-classified into model-based and data-driven. Filters and Observers are again identified as the commonly used model-driven methods.

9.6.4. Internal Model Based Safety

The concept of internal models used in this chapter is discussed in detail in chapter 3. Here, two properties of internal models are especially important: internal models can on the one hand be used as predictors enabling a robot to internally „try out “options before executing one. This can prevent dangerous situations and help to estimate the reaction of others to some action. On the other hand, internal models, more precisely coupled forward-inverse models, can also be used to recognize behavior of others. Specifically, in this context, the prediction error of these models, when trying to recognize behavior, can also be used to identify unknown and/or malicious behavior. This ability of an architecture based on internal simulation to detect faulty robots has recently been proposed and demonstrated experimentally by Millard et al. (2014b,a). The simulator incorporated the robot as well as the environment into the internal simulation.

This idea and the corresponding architecture of how internal models could lead to self-awareness and in turn enhanced safety is presented by Winfield (2014). This architecture will be discussed more detailed in chapter 10. In short, the architecture simulates possible future actions and evaluates their consequences in regard to some given safety criterion. A robot using this architecture is thus cognizant of the consequences of its actions and can therefore be called minimally self-aware; since the architecture only reduces the number of possible actions so the robot using it cannot be less safe than not using it. Furthermore, Winfield argues that the architecture itself, specifically the part actually evaluating the consequences might be checked using standard methods.

In contrast to most learned internal models, the internal simulator used in this architecture is capable of simulating not only the robot using the architecture itself and other robots as well as the environment, but more importantly also interactions between all these components. This includes interaction between robots and the environment and also interactions between robots. Even though learned internal models have been successfully used to recognize the action of other similar robots (Demiris and Khadhouri, 2006), these models are not capable of dealing with interactions yet. So using an internal simulator able to deal with robot-robot interactions lends itself very well to swarm-like scenarios, which are indeed mainly defined by the interaction between the different swarm members, either mediated through sensory inputs, i.e., sensing each other, direct physical interaction or through the environment, for example in foraging scenarios, which are the building block for all swarm algorithms.

One of the first experiments using self-simulation was demonstrated by Vaughan and Zuluaga (2006), who utilized an internal simulation modeling the robot as well as its environment to simulate possible future actions before executing them in order to avoid unsafe actions. In contrast to the architecture proposed by Winfield (2014), it only models the robot and its environment and not other agents and the result is the „best “action, which is to be executed in contrast to the actual consequences and their corresponding safety.

As discussed in chapter 3, internal models are a very promising candidate building block of a lot of sensorimotor and social skills as well as potentially for a theory of consciousness. They can, and have, been employed in various tasks besides safety as discussed in this section. Thus, in contrast to other approaches discussed here, the concept of using internal

models for a safety architecture is an holistic approach. This also means that it might be possible to re-use already existing internal models. This makes it one of the most promising approaches to increase safety and reliability of swarms.

Chapter 10

Internal Model Based Consequence Engine

The abstract architecture discussed in this section has been presented in (Winfield, 2014) and (Winfield et al., 2014) and was not devised by the author. The implementation of this architecture and measurements presented here were however performed by the author (for details refer to section 1.5).

10.1. Introduction

This chapter discusses an internal model based architecture as a solution to issues of swarm robotic systems presented in chapter 9. This discussion identified partial failures and/or malicious agents as one of the major problems in swarm robotic systems. Winfield (2014) proposed an internal model based architecture as a possible solution for these issues by making a robot cognizant of the consequences of its own actions. The internal model contained in this architecture basically consists of an internal simulator capable of simulating the robot itself, all other agents — robots or possibly humans — and the environment. This notably also includes interactions between agents, which makes this architecture particularly interesting for dealing with swarm-like scenarios, which are defined by the interactions between the swam members (or in this case mobile WSN nodes). The robot can then use this internal model to gain knowledge about the consequences of possible future actions without actually executing these actions. Using this knowledge, the robot can then avoid unsafe situations when interacting with other robots or possibly humans as demonstrated in chapter 12. A concrete toy example scenario of how this architecture works in the context of „ethical " decisions is presented in chapter 11. Furthermore, this internal simulator — more specifically its prediction error — can also be used to identify failed or malicious agents. A proof-of-concept experiment is implemented in section 13.2 and similar work was also presented by Millard et al. (2014a).

10.2. Architecture

The architecture discussed here is called Consequence Engine (CE) (Winfield, 2014; Winfield et al., 2014). A schematic view of this architecture is depicted in fig. 10.1. This architecture basically enables the robot to evaluate the consequences of its next possible actions, which

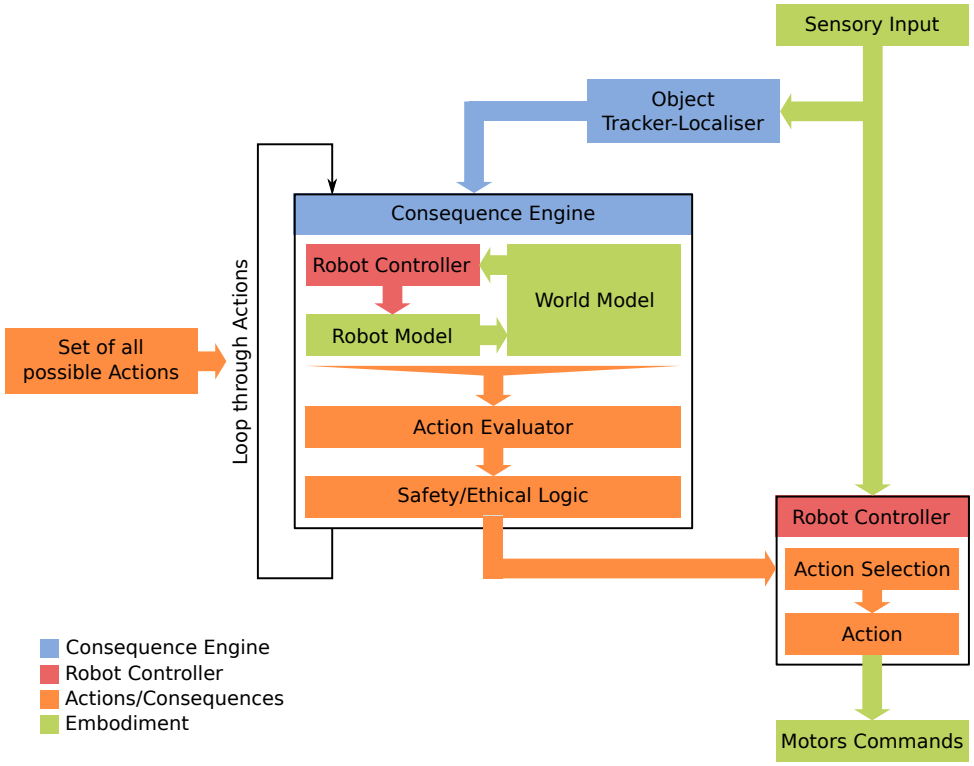


Figure 10.1.: The Consequence Engine (CE) architecture. Note that the module performing the simulation and inferring the consequences of the different actions is also called CE. Modules dealing with different kinds of information are color coded as specified in the legend.

are executed inside of the simulator, which is contained in the CE and uses an internal model[1], instead of actually executing it. The consequences resulting from each action are evaluated respectively. This process is relatively independent of the robot's controller in that the CE only provides estimates of the consequences of the next possible actions but the robot itself chooses and then executes the next action in the real world. The CE thus only acts as a kind of advisor.

The CE architecture consists of several modules, which are discussed in detail in section 10.3.3. First an overview of the interplay of the modules as depicted in fig. 10.1 is given. The respective modules are presented in detail in the next section 10.3. The CE is fed with information about the current situation by the Object Tracker-Localiser (OTL) and then loops through all possible next actions. In the simplest case, this set of actions is fixed by design but it could as well be dynamically generated. For each candidate action the CE simulates the robot itself executing the action, as well as all other entities tracked by the OTL and

[1]Depending on the point of view, the simulator could also be described to contain multiple paired inverse and forward models, one for each entity (robot, environment, etc.) and also one for each possible controller of the robots (see chapter 3).

described by the internal model as well as the environment. It therefore generates a set of simulation outputs, for example trajectories of all agents, which are then evaluated by the Action Evaluator (AE). The evaluation of the physical consequences performed by the AE are then passed to a separate Safety/ethical Logic (SEL) module implementing the definition of safety and/or ethics as defined by the specific task. The details of AE and SEL are discussed in detail in section 10.3.2. Once the CE has generated a complete set of consequences and their evaluations, it is passed to the Action Selection (AS) mechanism of the robot, which then selects one of the actions according to this evaluation for execution by the robot controller.

10.3. Consequence Engine Details

10.3.1. Set of Actions and Corresponding Safety/Ethical Values

The safety values consist of two contributions: the base value and the safety/ethical value. The base value encodes the task of the robot to be followed if no danger is imminent. The safety/ethical value then modifies the base value according to the SEL if the CE detects imminent danger but yields no contributions otherwise.

As depicted in fig. 10.1, the set of all possible actions is something external to the process of the CE. Basically, these actions, together with their basic values, describe the task of the robot. The rules by which the SEL modifies these basic values then defines the „ethical “ rules which the robot is to conform.

One way to create this set of actions and corresponding base values in the context of the robot controllers discussed in section 10.5.2 is to discretize the relevant space into a grid and assign base values to the grid points according a scheme based on PFs encoding the task of the robot. These grid points then are points which the robot can move towards. This does not imply a discrete motion since the robot can still move continuously through space but can only stop at those points. This constructed PF is sampled at the grid points and these values are used as the base values. The PF then encodes the goal of where the robot is to move towards and also the preferred way of getting there, following the gradient of the PF.

The consequences of all possible actions are simulated and evaluated by considering the predicted trajectories for each robot. The base safety values are then modified according to the task-specific safety definition by the SEL. The AS of the robot controller then picks the best possible action according to the safety values. Usually, this action is the one which gets the robot closest to its goal defined by the PF while still being safe.

10.3.2. Action Evaluator, Safety Logic and Action Selector

The simulator provides the AE with trajectories for all robots during the simulated future times. The AE then checks the supplied trajectories against the safety condition as specified by the task at hand. The SEL then severely discourages the actions failing that condition by modifying the safety value, effectively pruning these actions from the set of possible next actions. If more complex decisions have to be made by the robot, the rules employed by the SEL can also be used to weigh several factors against each other. In principle, this step of the

SEL modifying the base values can also be performed before discretizing space by modifying the PF directly.

For example, the safety condition employed in chapter 12 and specified in section 12.2 states that an action is considered to be safe if no other robot is closer to the robot than some safety distance at all times. For the AE this simply means checking the trajectories of all simulated robots for every simulated time step against this condition.

The AS then chooses the action with the best safety value. This effectively means that it follows the gradient of the task-specific PF while avoiding dangerous situations. Note that the AS is part of the robot controller and the robot can thus also choose to select actions with a worse safety value than the safety value of the best action according to the SEL.

10.3.3. Object Tracker-Localizer and Models

The OTL has been implemented using an off-the-shelf motion capturing system and outfitting the robots with the corresponding markers. This system is discussed in more detail in section 10.5. In principle this module could also be implemented on the robot using a camera and computer vision algorithms or by all agents broadcasting their own position measured for example by a self-localizing system.

The internal model is implemented using a slightly modified off-the-shelf simulator (see section 10.4.1) and is discussed in detail in section 10.4. All models of other robots and the environment are hard-coded, i.e., known a priori, to simplify the experiment in order to be able to focus on the architecture itself. In principle, all models, i.e., world model as well as model for other entities etc., could also be based on a learned internal model. Initial experiments for doing this are shown in chapter 13.

10.4. Internal Simulator

For implementing the internal model used by the CE, an off-the-shelf simulator was chosen. In general, simulations are used in robotics, often using advanced physics and sensor models, to test and develop robots and the corresponding controllers before they are implemented in hardware to reduce development time and also the risk associated with bugs, faults and failures. Examples of standard robotic simulations include Gazebo (Koenig and Howard, 2004), Webots (Michel, 2004), Player-Stage (Vaughan and Gerkey, 2007) and Morse (Echeverria et al., 2011). When using a simulator, the so-called reality gap (Jacobi et al., 1995), created by the approximation of the performance of real sensors and actuators by the simulator, i.e., consequently by the respective sensor and motor model, has to be taken into account. This topic will be further discussed in section 10.4.3.

10.4.1. The Stage Robot Simulator

The simulator running the internal simulation is a modified Stage simulator (Vaughan and Gerkey, 2007; Vaughan, 2008). It was modified to act as simulation server in a service-oriented architecture accepting `SimRequest` messages (see fig. 10.5) over a network connection. This service-oriented architecture was chosen as it allows in principle to scale the

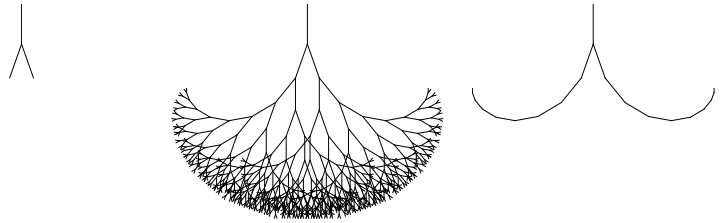


Figure 10.2.: Search trees for a fig. 10.2a one step search, fig. 10.2b full search, fig. 10.2c depth-only search

simulator performance to large numbers of `SimRequest` requests by using a transparent load balancer acting as a virtual simulation server, which redistributes the requests to a number of identical actual simulation servers. This approach takes the fact into account that the loop performed by the CE over the set of possible next actions is embarrassingly parallel. These requests consist of a number of robots, their initial poses, the configuration for their controllers, and the simulation time. Upon execution, the server returns complete trajectories for all robots for the simulation time.

The simulation contains a copy of the environment, the robots used, their sensors and their controllers. In most experiments, the controllers execute exactly the same code as the real robots and the sensors and motors are calibrated to the real ones empirically. In principle learned robot models and controllers could also be used. The performance of the simulation and the resulting simulation budget is discussed in section 10.4.4.

10.4.2. Decision Search Tree

Most internal models based architectures in robotics perform only one time-step into the future, especially when dealing with learned body models (see section 3.2.3 for approaches to go further than one time step). In this work, the goal is to simulate much further — possibly several orders or magnitude – into the future then one time-step employing the speed and precision of the off-the-shelf simulator and full knowledge of the world.

As depicted in fig. 10.2, these multi-step simulations lead to search trees if the robot can perform more than one action, which is naturally the case for the CE architecture as it specifically deals with a set of possible next actions, since the robot can in principle decide at each time step to change the current action. Performing simulations of only one step and doing depth-only search are two extremes of traversing the full decision tree. Searching a full decision tree is not practical because of the curse of dimensionality, so some sort of pruning has to be devised especially for deeper searches. In contrast, choosing a depth-only search as used here constitutes one of the most simple and straight-forward ways of accomplishing this goal. Using depth-only search is a valid simplification if the robot controllers of the other robots in the simulation are stateless (see section 10.5.2).

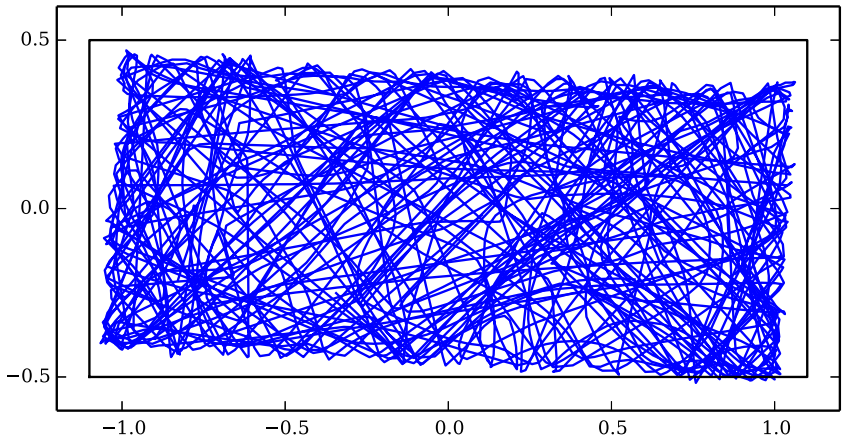


Figure 10.3.: Trajectories for the simulation error calibration

10.4.3. Reality Gap: Simulation Error Measurements

There is a reality gap between every simulator and the physical object it is to represent. The main reasons for this gap are: numerical errors of the simulation, approximation errors of the underlying mathematical models, calibration errors of the model parameters against the real systems, and measurement errors of the initial conditions of the simulation. To measure this gap experimentally, six robots were used in the arena described in section 10.5 executing the simple action `GoStraight(0.8);Avoidance`. Their trajectories were recorded for $489\,\mathrm{s}$. The resulting real trajectories are depicted in fig. 10.3.

From the mismatch between the trajectories and the arena walls, it is obvious that the calibration of the motion capturing system placed its coordinate system with a rotation of about 3°, and also with a slightly offset center in relation to the coordinate system of the simulation. This is mainly due to the calibration process of the motion capturing system using a manually placed ground plate with markers to establish the motion capturing coordinate system.

Simultaneously, the movement of the six robots was projected into the future via simulation for different simulation times. After finishing the experiment, the error between the simulated, i.e., prospected, trajectories and the actual real trajectories were calculated for all time steps and simulation times. The results are depicted in fig. 10.4.

The figure shows normalized errors, which means that the cumulative error is divided by the length of the real trajectory. Due to this, there are numerical errors for small simulation times as the measured error is divided by a small number, so the increased error for short simulation times is a purely numerical artifact. For longer simulation times, there is a linear increase in the median error, which is most probably due to the mis-alignment between the motion capturing coordinate system, which is the same as used for the simulation, and the real coordinate system. This error is far larger than the one caused by inaccuracies of the simulation, which are expected to be scaling roughly quadratically in time.

In general, simulation errors are mitigated to some extent by the fact that the CE, which is

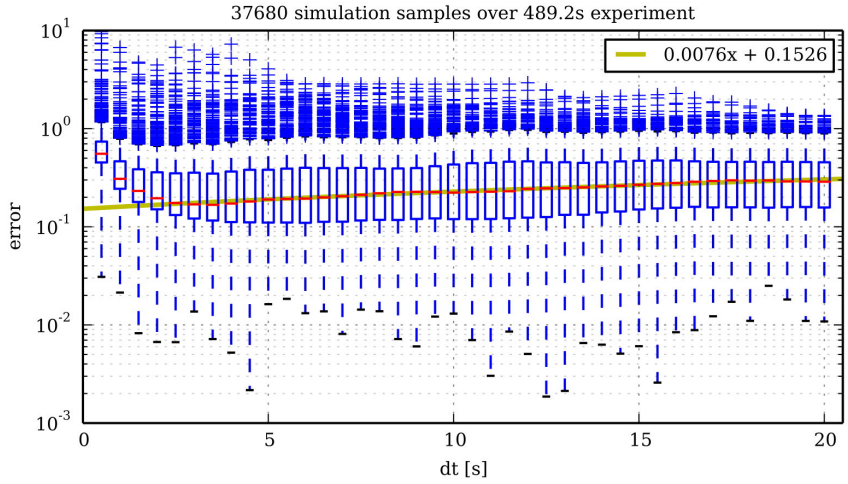


Figure 10.4.: Simulation error as a function of simulation time

running at 2 Hz, is periodically updating the simulations it performs by new simulations in a memoryless fashion, discarding old simulation results. As seen in fig. 10.4, the simulation error is the smaller the shorter into the future an event is occurring. This means for example that the closer two robots get, the more accurate the predictions of a possible interaction becomes.

The virtual sensors of the simulated robot, in particular the IR sensors, have to be calibrated to the real sensors. This calibration was performed with a focus on reproducing the behavior of the real robot, i.e., turn rates and radii when performing avoidance behavior close to walls and other robots. However, the particular calibration employed here led to the effect of simulated robots getting stuck on walls or in rare cases on other robots[2]. This effect only occurs only for long simulation times and only rarely. In real experiments, measurement errors and motor, i.e., movement, noise of the real robots mitigates this effect completely. For pure simulations in contrast, i.e., simulated robots using a CE to internally simulate other simulated robots, this effect can be observed in 10 % to 20 % of experiments. This effect is only relevant for the simulation study performed in section 12.4.3 as all other experiments were conducted with real robots. For this study the simulation runs in which this effect occurred were discarded while performing the statistical analysis so they have no effect on the statistical validity of this study.

10.4.4. Simulation Budget

The CE runs at approximately 2 Hz so there is a budget of 0.5 s to loop through all possible actions, simulate their outcome, asses the consequences, assign a safety value to each action and then choose the best possible action. In relation to the computational power nec-

[2]This effect is probably due to the response of the virtual IR sensor being to weak. However, it proved impossible to at the same time reproduce correct trajectories and eliminate this effect.

essary for the simulation, the other tasks are negligible. For simplicity, they are therefore not considered for the analysis.

As mentioned in section 10.4.1, simulation time runs at about 600 times real time on the used hardware configuration[3], so $300\,\mathrm{s}$ can be simulated effectively during one update cycle. This simulation time is the simulation budget, which has to be allocated to the different possible next actions.

Considering the maximum speed of an e-puck robot of about $0.1\,\mathrm{m\,s^{-1}}$, simulation times should be in the range of about $10\,\mathrm{s}$, which corresponds to $1\,\mathrm{m}$ traveled distance at maximum speed, for the robot to gain meaningful new information. This means that the simulation budget is about 30 different future actions.

As discussed in section 10.3.1, the arena used in most of the experiments ($2.2\,\mathrm{m}\times1.8\,\mathrm{m}$) is discretized into a grid of for example 6×5 points to generate the set of possible actions. Assuming a simulation time of $10\,\mathrm{s}$ this already completely saturates the simulation budget. This is the reason why sometimes attention mechanisms (see section 12.3.1) and adaptive simulation times (see section 12.3.2) heuristics have to be resorted.

10.5. Experimentation-Centered Implementation

Figure 10.5 shows the flow of information between the different modules of the implementation of the architecture described in section 10.2. There are some organizational differences to the abstract architecture (compare fig. 10.1).

The main difference to the abstract architecture is the high level controller. This module contains the CE, the SEL and the AS of the intelligent robot as well as all the logging, book-keeping and experimentation tools. It is very centralized for the sake of experimentation efficiency, i.e., mainly the logging of trajectories, safety values and decisions. Besides performing the role of the CE it also manages the automated experimentation, so it can also control all the robots used in the experiment. This feature is only used between experiments for set up, for example new initial conditions. By changing the control actions for the low level controller (see section 10.5.2) it can effectively remote-control the robots. In fig. 10.5 this mechanism is represented by the flow of `CtrlRequest` messages from the high level controller to the robot controllers. While experiments are run, the different robots act completely independent of the high level controller. Only the intelligent robot is influenced by the outcome of the AE.

However, this implementation choice does not constitute a functional difference to the abstract architecture and could as well be implemented on the robots themselves (with some complications for the logging system).

Thus, the high level controller, i.e., the CE, and the simulation server described in section 10.4, are run on the same computer. The high level controller of the architecture, i.e., the consequence engine, is implemented in Python. The low level robot controllers and the simulation are implemented in C/C++.

[3] Lenovo T440s, Intel i5 4200 @ 1.6 GHz, 12 GB RAM

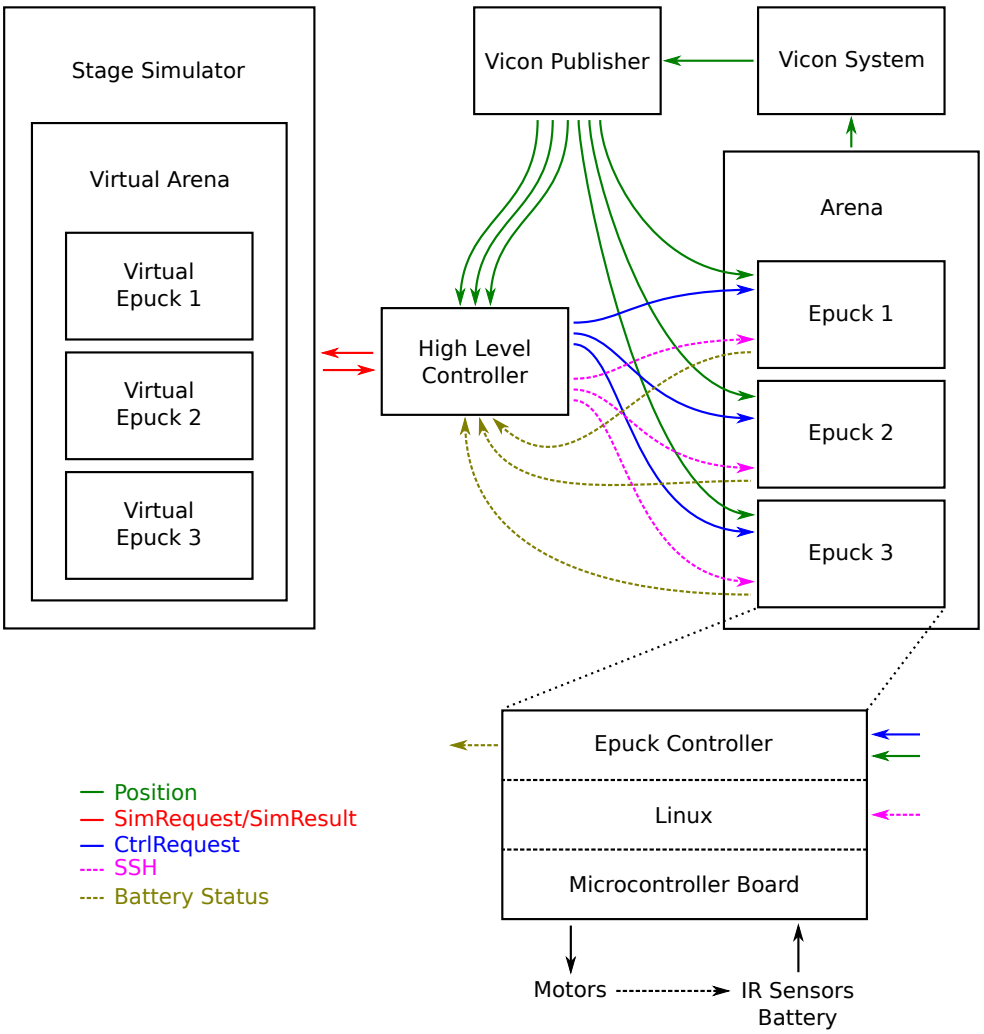


Figure 10.5.: Flowchart of the complete experimental system including all the logging, bookkeeping and status messages used for experimental control and logging. This diagram not only contains all the elements needed for the CE but also all elements needed for automated experimentation so there are more control- and logging messages than would be needed for the operation of the CE alone.

Figure 10.6.: Experimental infrastructure showing the arena and the Vicon tracking system

10.5.1. Infrastructure and Physical Setup

Figure 10.6 depicts the physical setup consisting of an arena of size $2.2\,\mathrm{m} \times 1.8\,\mathrm{m}$, a motion capturing system and an overhead camera.

The off-the-shelf motion capturing system [4] runs at $50\,\mathrm{Hz}$, implements the OTL and broadcasts the positions of all robots via simple UDP messages to all participants of the wireless network at a reduced rate of $15\,\mathrm{Hz}$ (see also fig. 10.5). The communication between the robots and the simulation server and between sub-components is based on the Google Protocol Buffers library [5] and executed via Transmission Control Protocol (TCP). All robots, the simulation server, the tracking system, and the logging system are connected via an `IEEE 802.11g` WLAN in infrastructure mode. Additionally, a video camera is recording the experiments for later analysis and for demonstrative purposes.

10.5.2. E-pucks and Controllers

E-puck mobile robots (Mondada et al., 2009), equipped with a Linux extension board (Liu and Winfield, 2011) were used as a mobile base. Additionally, optical markers were arranged in individual patterns on top of the robots to facilitate tracking with the motion capturing system. Two robots are shown in fig. 10.7.

The robots are equipped with infrared sensors, which are used for basic obstacle avoidance, a camera, which is not used, and virtual sensors to sense their own pose and those of all other

[4] Vicon Motion Systems Ltd., `http://www.vicon.com/`
[5] `https://code.google.com/p/protobuf/`

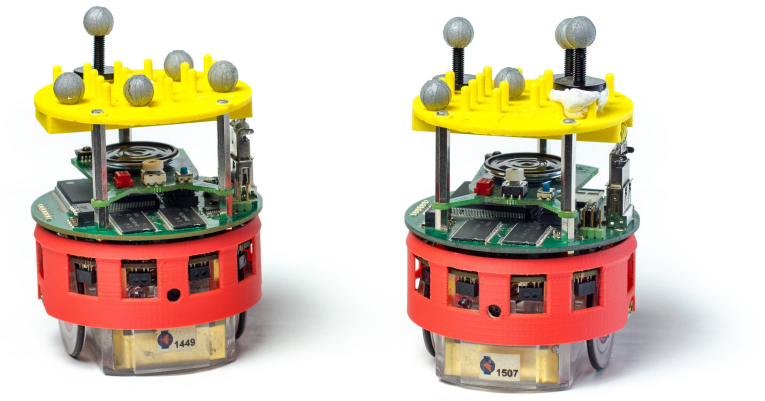


Figure 10.7.: Two e-puck robots with markers and IR sensors. The e-puck with Linux board fitted in between the e-puck motherboard (lower) and the e-puck speaker board (upper). Note the yellow „hat “, which provides a matrix of pins for the reflective marker spheres, which allow the tracking system to identify and track each robot.

robots. These virtual sensors are simulated making use of the broadcasted position messages of the motion capturing system.

All robots run a stateless controller with a fixed set of sub-actions. Those sub-actions are:

- `GoStraight(speed)`: move straight with a maximum `speed` of $1\,\mathrm{m\,s^{-1}}$
- `Avoidance`: Braitenberg style (Braitenberg, 1986) avoidance using IR sensors
- `MoveTo(x,y)`: move to coordinate `(x,y)` using the virtual global position sensors
- `Stop`: do nothing

Additionally, there are sub-actions not used during experiments but for the automated experimentation for tasks like for moving robots into initial poses or related to hardware calibration and debugging etc.:

- `TurnLeft(speed)`: turn left with `speed`
- `TurnRight(speed)`: turn right with `speed`
- `Follow(name, distance)`: follow robot with `name` at `distance` (this sub-action is only used in section 13.2 and is also described in more detail there)
- `CalibrateIR`: calibrate IR sensors
- `ResetDSPIC`: reset the basic robot microcontroller board
- `PrintProximityValues`: print the values measured by the IR sensors to `stdout`

Actions are composed of a number of concatenated sub-actions and are executed at 10 Hz by the robots, independently of each other and independently of the CE. These actions form the vocabulary used for the set of all possible actions. The simulated robots in the CE run exactly the same code as the real ones, also at 10 Hz in simulated time, i.e., faster than real time.

10.5.3. Experiment Logging and Analysis

Even though the presented implementation uses a centralized high level controller as presented in section 10.5 for logging purposes, the actual logging messages are generated asynchronously by the individual robots. The arriving messages are time-stamped by the central clock of the high level controller but the resulting time series are neither synchronized nor truly periodic due to delays and experimental noise. The python package pandas McKinney (2011) is used to interpolate these different time series to the same periodic index for analysis. This is necessary as most comparison metrics like for example Euclidean distances are not well defined for unsynchronized aperiodic time series.

This work was presented in (Winfield et al., 2014)[1].

Chapter 11

Towards an Ethical Robot

11.1. Introduction

The idea of robots interacting with humans and the possible consequences of these encounters have already been extensively addressed by science fiction authors as a hypothetical question long before elaborate robots became a reality. Asimov (1950) already envisioned ethical principles for human robot interaction encoded in his „Laws of Robotics “. These early ideas have by now become an active research area and the general consensus at the time of writing is that robots should be more than just safe in the conventional sense. For example Wallach and Allen (2009, p. 17) write

> If multipurpose machines are to be trusted, operating untethered from their designers or owners and programmed to respond flexibly in real or virtual world environments, there must be confidence that their behavior satisfies appropriate norms. This goes beyond traditional product safety. Of course, robots that short-circuit and cause fires are no more tolerable than toasters that do so. However, if an autonomous system is to minimize harm, it must also be „cognizant “ of possible harmful consequences of its actions, and it must select its actions in the light of this „knowledge “, even if such terms are only metaphorically applied to machines.

As discussed in chapter 9, internal models are a good candidate mechanism to make robots and multi-robot systems safer and more reliable. This is especially important when interacting with humans, where robots have to be more than just safe. In this chapter an application of the CE discussed in chapter 10 to this issue is presented. Using the mechanism of the CE a robot implements an „ethical “ action selection mechanism, which can lead to the robot compromising its own safety and the task objectives to prevent a second robot to come to harm. This then leads to the robot not only being safe but also ethical.

Such a robot is cognizant of the consequences of its own actions not only on itself but also on other entities, for example other robots. Thus this experiment is — besides being interesting in itself because of the ethics aspect — a demonstration of the general ability of

[1]For a detailed breakdown of the contributions of the different authors refer to section 1.5.

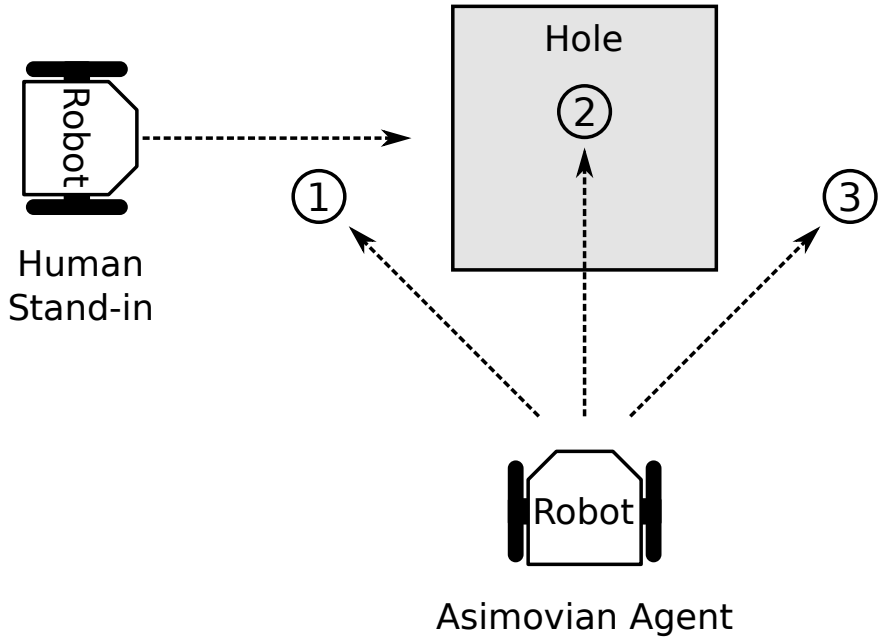


Figure 11.1.: A scenario with both safety and ethical consequences

the CE to enable a robot to reason about the collective consequences of its own actions in a multi-robot or possibly swarm scenario.

Using the architecture of the CE described in chapter 10 a robot is outfitted with an ethical action selection mechanism. Experiments then show that the robot can thus sometimes compromise its own safety and the task objective to perform the more „ethical “ action to prevent another second robot to come to harm. First a simplified example for this strategy is given and a way to implement „Asimovian “ ethics is shown. Then several experiments with one to three robots are presented.

11.2. Robots with Internal Models

11.2.1. A Toy Example Situation

This hypothetical toy experiment was presented by Alan F. T. Winfield (Winfield, 2014) as an abstract example to demonstrate how the CE, which also was first presented there, works. However it was not implemented experimentally. In section 11.3 a real implementation and the corresponding experiments of this toy experiment are presented.

Consider an example situation consisting of two robots, robot A (A for Asimovian) and robot H (H for human stand-in), and a hole in the ground. This situation is depicted in fig. 11.1. Robot A is tasked to move to the right side of the hole while robot H is tasked to go straight in the direction of the hole.

Robot A can choose between three actions: (1) to go ahead left, (2) to go straight ahead and (3) to go ahead right, i.e., to its tasked destination. If robot A chooses action (1), it will

Robot action	base value	robot A consequence	robot H consequence	total value
(1) Ahead Left	-10	0	0	-10
(2) Ahead	0	-1000	-100	-1100
(3) Ahead Right	10	0	-100	-90

Table 11.1.: Safety outcome values for each robot action, for the scenario depicted in fig. 11.1

possibly collide with robot H but robot H will try to avoid robot A and thus be prevented from falling into the hole. If robot A chooses action (2), it will, as well as robot H, fall into the hole. And if robot A chooses action (3), it will reach its tasked destionation and avoid the hole but robot H will fall into the hole.

These consequences can be encoded in safety values as discussed in section 10.3.1. The actual values and their different components are listed in table 11.1. The base value of the safety value encodes the task description for robot A, i.e., to go the right side of the hole with values decreasing the further the actions lead the robot from the goal. The consequences for robot A heavily discourage it from falling into the hole and the consequences for robot H discourage any actions that lead to robot H falling into the hole. Note that the consequences for robot A are more heavily weighted than for robot H[2].

These safety values encode what is called the SEL in the CE (see section 10.2 for details). Instead of stating the safety values, this can also be written in a logic form:

```
IF for all robot actions, the human is equally safe
THEN (* default safe actions *)
    output safe actions
ELSE (* ethical action *)
    output actions for least unsafe human outcomes
    which are still safe
```

This set of rules is already remarkably close to what Asimov (1950) envisioned in his „Laws of Robotics “. Even though this minimalistic model works in this toy scenario, real world examples are almost always more complex. Designing a practically viable version of the SEL will probably require the help of ethicists.

11.2.2. Real World Safety Outcome Values

A simplistic example of how the AE can evaluate the consequence of actions was described insection 11.2.1. For simplicity, the example only consists of three possible actions, tailored to fit the exemplary situation described. In a real robot the simulation budget (see section 10.4.4) can be used up completely to evaluate more than just a minimal number of tailored actions in order to generate more flexible and adaptable robot behaviors.

Actions are generated by discretizing the space needed for the experiment into a grid of points to which the robot can move. Trivially the whole arena could be discretized but

[2]This is the main difference to Asimov's original Laws of Robotics.

simulating all these actions would exceed the simulation budget, so a smaller area around the virtual hole and the goal was chosen. Specifically, an area of $1\,\mathrm{m} \times 1\,\mathrm{m}$ was discretized into a 6×5 grid of points, some of which fall inside the virtual hole.

For dealing with a larger number of actions, an algorithmic way to calculate safety outcome values for all those actions is needed. For this, a good option is to choose the paradigm of virtual PFs. Here one PF which drives Robot A towards its goal, similar to the first column in table 11.1, is employed . Another, stronger PF is employed if the simulation shows danger for one of the other robots and favors actions which move robot A towards the robot in danger. This second PF is only employed when danger is imminent and is zero otherwise (this PF is not strictly necessary but significantly improves the reaction times of robot A). The sum of these PFs is sampled at the grid points and assigned as basic safety values to the actions.

No additional penalty is placed on getting too close to other robots during normal operation since the robots' real IR sensors and controllers are used for basic collision avoidance. If this aspect were to be included, a PF could be used to discourage areas close to other robots.

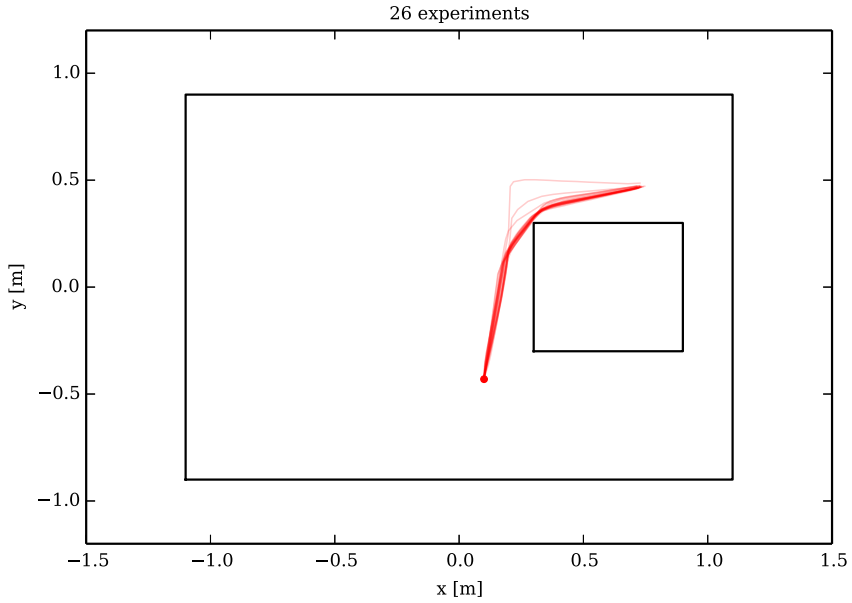
After assigning the basic safety values, robot A's SEL considers the estimated danger for all robots generating effectively the equivalent to the second column of table 11.1. Danger for robot A is severely penalized and danger for the other robots is also penalized, but less severely, as in the example in table 11.1. The values of the penalties were chosen to conform with the general structure outlined in section 11.2.1.

11.3. Experiments

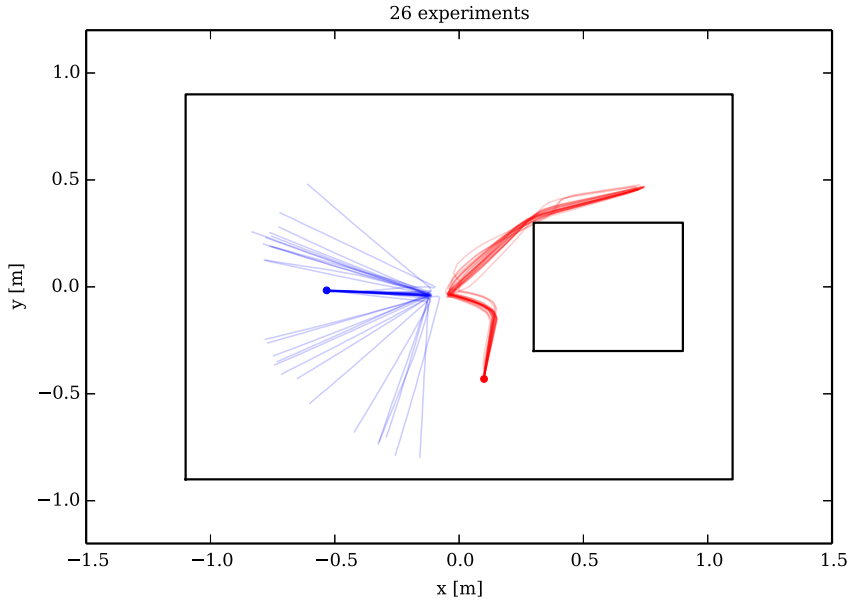
A set of three experiments was conducted in the spirit of the basic experiment outlined in section 11.2.1. The first experiment consists of robot A, which is running the CE only, navigating a safe path to its goal destination while using its CE system to safely avoid the hole in the arena. This experiment provides a baseline test in which robot A only has to ensure its own safety. The second experiment adds robot H, acting as a proxy human, to test the ability of robot A to model both itself and H, and if necessary deliberately interacting with H in order to prevent it from falling into the hole. A third experiment adds a second proxy human robot H2 in order to present A with a dilemma: can it prevent both H and H2 from coming to harm?

11.3.1. Experimental Setup

The general experimental setup of the CE and experimental environment is described in section 10.5. The scenario shown in fig. 11.1 is implemented experimentally by creating a virtual hole in the ground, of size $0.6\,\mathrm{m} \times 0.6\,\mathrm{m}$ in the full arena of size $2.2\,\mathrm{m} \times 1.8\,\mathrm{m}$. This virtual hole is sensed only by robot A's virtual sensor, a second robot (H), and later a third robot H2, are additionally introduced into the arena. Robot H does not have the internal-modeling architecture of robot A. It has a simple control system allowing it to move around the arena in straight lines, avoiding obstacles with its infra-red proximity sensors, but lacking the virtual sensor of robot A it is unable to „see " the virtual hole in the arena.

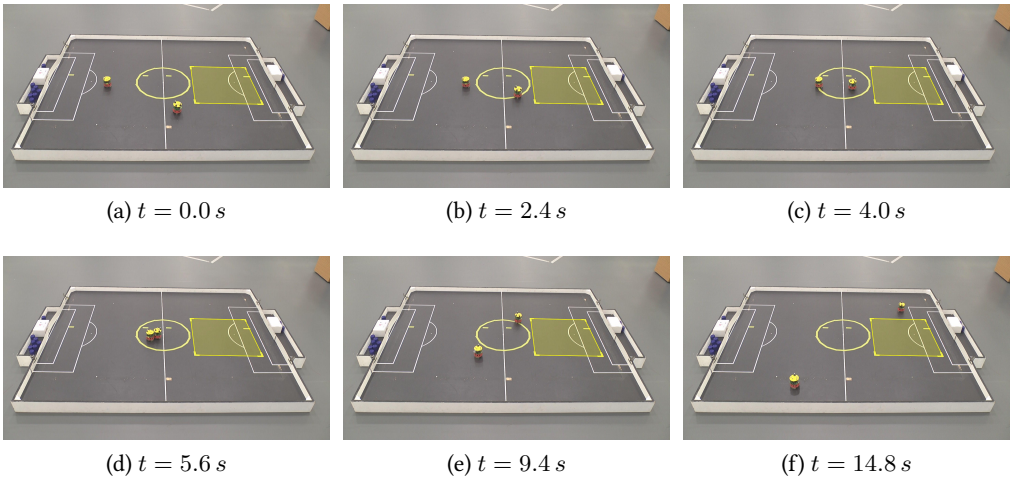


(a) Experiment 1



(b) Experiment 2

Figure 11.2.: Superimposed trajectories of robots for experiment 1 and 2 respectively. Robot A is shown in red, with start position on the left and goal on the upper right; robot H is shown in blue with start position in the lower center of the arena.



(a) $t = 0.0\,s$ (b) $t = 2.4\,s$ (c) $t = 4.0\,s$

(d) $t = 5.6\,s$ (e) $t = 9.4\,s$ (f) $t = 14.8\,s$

Figure 11.3.: Frames from experiment 2: (11.3a) start (11.3b) Robot A starts normal operation and moves towards its goal. Robot H starts moving towards the hole. (11.3c) The CE of A detects danger for H and moves to intercept it. (11.3d) A intercepts H. (11.3e) Danger for H is averted and A continues towards its goal. (11.3f) A reaches its goal.

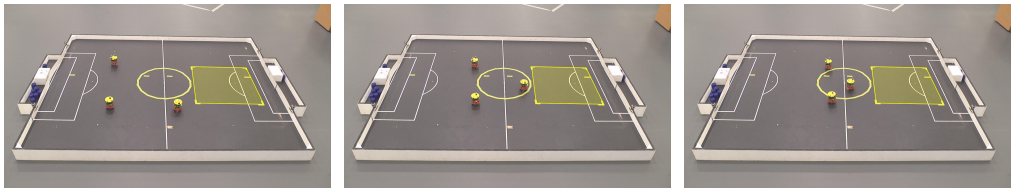
Robot A's virtual sensors allow it to both see the hole in the arena and also track the position and direction of motion of robot H. Robot A is therefore able to initialize its CE with both its own position and heading, and the position and heading of robot H.

As discussed in section 10.5.2, the robots run stateless controllers with a fixed set of sub-actions. The sub-actions used in the experiments are: `GoStraight(speed)` with a maximum speed of $1\,\mathrm{m\,s^{-1}}$, `Avoidance` for Braitenberg style (Braitenberg, 1986) avoidance using IR sensors and `MoveTo(x,y)` using the virtual position sensors also with a maximum speed of $1\,\mathrm{m\,s^{-1}}$. Actions are composed of concatenated sub-actions and are executed at $10\,\mathrm{Hz}$ within the robots, independently of the CE. The action of robot A is determined by the AS after considering the safety values generated by the CE. The set of possible next actions over which the CE loops is fixed for all experiments and consists of `MoveTo(x,y); Avoidance` with `(x,y)` values according to the grid defined in section 11.2.2. Robot H runs the action `GoStraight(0.7); Avoidance` for all experiments.

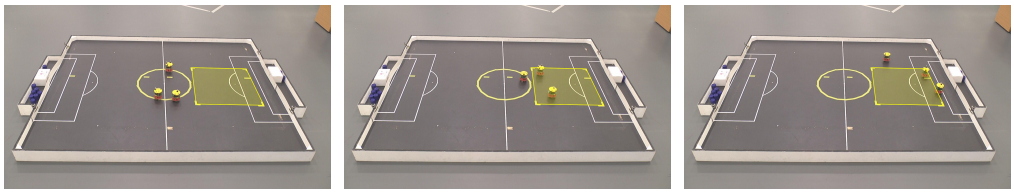
For all experiments, robot A had prior knowledge about the controllers robots H and H2 were executing, i.e., of their actions. In principle this could also be learned from their trajectories (see section 13.3), but for simplicity and since it was not the focus of this experiment, prior knowledge was chosen.

11.3.2. Experiment 1: Baseline with Robot A only

In this experiment the safety values consist only of the original PF driving robot A towards its goal. The starting position and goal are chosen in such a way that the unmodified PF, which is proportional to the distance to the goal, would drive robot A straight into the virtual hole.



(a) $t = 0.0\,s$ (b) $t = 5.7\,s$ (c) $t = 7.6\,s$



(d) $t = 10.5\,s$ (e) $t = 14.8\,s$ (f) $t = 21.2\,s$

Figure 11.4.: Frames from experiment 3: (11.4a) Initial conditions with H and H2 pointing towards the hole. (11.4b) A detects danger for both H and H2. (11.4c) A cannot decide which of the robots to rescue. (11.4d) A misses the chance to rescue either robot. (11.4e) A turns around to continue towards its goal since it's now too late to rescue the other robots. (11.4f) Robot A reaches its goal.

The CE then evaluates all possible actions and penalizes the ones driving robot A into the hole, effectively guiding it around the hole. Overlaid trajectories for this experiment are shown in fig. 11.2a and show that robot A is able to avoid falling into the virtual hole, with 100% reliability.

11.3.3. Experiment 2: Robots A and H

This experiment is an extension of the first one with the same goal and initial condition for robot A. To demonstrate the effectiveness of the CE approach, the second robot H was added, as described in section 11.2.1. The controller of robot H is executing the simple action (see section 10.5.2) `GoStraight(0.7); Avoidance` and initial conditions which point it directly towards the virtual hole. Successive snapshots of a typical experimental run are shown in fig. 11.3.

The run starts with robot A following the same trajectory as in the first experiment, but as soon as the predictions of the CE for robot H shows that H would fall into the hole if not intercepted, robot A diverts from its normal trajectory to intercept and thus „rescue “ robot H. Robot A then continues back onto its original trajectory and reaches its goal.

Figure 11.2b shows trajectories for a number of experiments. In all cases robot A succeeds in rescuing robot H by intercepting and hence diverting robot H. The beginning and end of robot A's trajectories are exactly the same as in the first experiment.

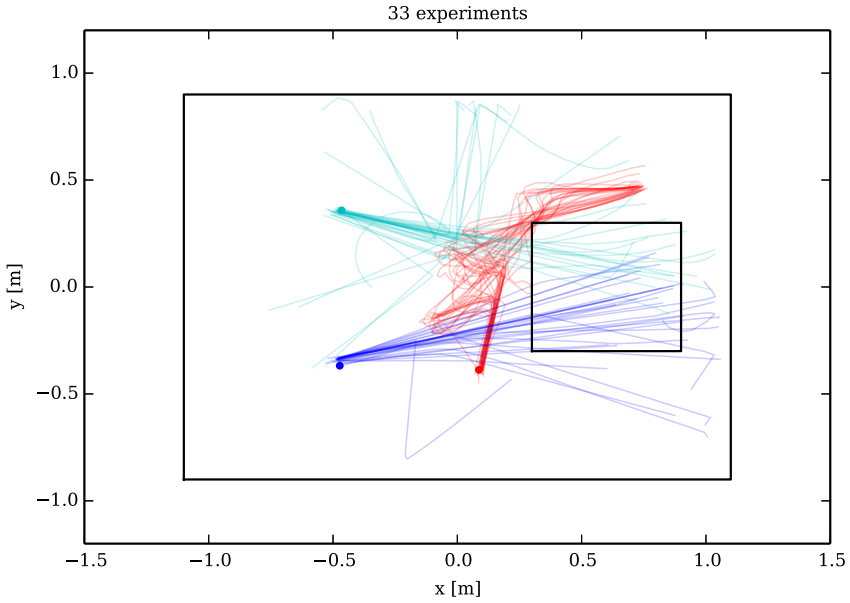


Figure 11.5.: Superimposed trajectories of robots for experiment 3. Robot A is marked in red, with start position on the left and goal on the upper right; robot H is marked in blue with start position in the lower center of the arena and robot H2 is shown in cyan with start position in the upper center of the arena.

11.3.4. Experiment 3: Robots A's Dilemma

In this third experiment, a third robot H2 is introduced, presenting robot A with the dilemma of having to decide which of H and H2 to rescue. Both H and H2 start pointing towards, and equidistant from the virtual hole (see fig. 11.4a), while the initial and goal positions for robot A remain unchanged.

Figure 11.4 shows successive snapshots for one experimental run. Robot A is unable to resolve its dilemma in this particular run since its CE does not favor either H or H2, which results in A trying to rescue both at the same time and failing to rescue either.

Trajectories over a series of 33 runs are shown in fig. 11.5. In 3 (9%) experiments two robots were rescued, in 16 (49%) experiments one robot was rescued, and in 14 (42%) experiments no robot was rescued by robot A. Surprisingly and perhaps counter-intuitively, robot A is able to rescue at least one robot in more than half of runs, and sometimes even both robots. The reason for this is noise. The robots don't start at exactly the same position every time, nor do they start at precisely the same time in every run. Therefore, sometimes robot A's CE indicates danger first for one robot and since the CE only runs at 2 Hz, robot A by chance rescues this robot. As soon as one robot is rescued, the experiment resembles experiment 2 and if physically possible, i.e., robot A has enough time left to react before the other robot reaches the virtual hole, it also rescues that robot.

This discussion also shows that in principle the dilemma is already latent in the second

experiment. If the dilemma is to be resolved, the symmetry of how the actions are evaluated for saving the different robots has to be broken. This in turn would mean that some of the robots have to be favored over other ones.

11.4. Conclusions

The CE architecture has been proven capable of to implementing a minimally ethical robot using the internal model/simulation paradigm. An instructive toy example scenario was discussed and subsequently also implemented with real robots. The experiments show that the robot running the CE is able to maintain its own safety and if necessary also to prevent other robots to come to harm.

The third experiment shows that even a minimally „ethical “ robot as implemented here can face a dilemma. However, this experiment also shows the surprising fact that in real world scenarios, those „latent “ dilemmas are often resolved noise inherent to real experiments. In general it would also be easy to implement a rule preventing the dilemma situation by breaking the symmetry between the other robots, i.e., by preferring some over others. Such a rule however should be decided by a collaborative effort between the engineers designing the robot and ethicists to ensure ethical grounding.

This work was presented in (Blum et al., 2015)[1].

Chapter 12

Internal Model Based Safety

12.1. Introduction

To demonstrate the effectiveness of the Consequence Engine (CE) approach, as introduced in section 10.2, for safety applications, an experiment involving an intelligent (using a CE) robot and several dumb, i.e., Braitenberg-style, robots was conducted. The intelligent robot is to move from one end of a corridor to the other end while avoiding the dumb robots moving around randomly. This is also an exemplary task for a robot moving through a human environment, for example an office corridor while physically avoiding humans moving in the area.

Besides the practical aspect of utilizing the CE to implement a safety application, this experiment also demonstrates that the CE can perform with larger numbers of robots than in the experiment presented in chapter 11, where only a maximum of two other robots had to be simulated. The number of robots in this safety experiment — five other robots plus the robot implementing the CE — is of the order of the number of nearest neighbors in a typical swarm scenario.

Furthermore, this experiment was designed to show how the CE can, by using its internal simulation mechanism, also take into account second order effects, i.e., robots interacting with the environment, which in this case first of all means interacting with the walls of the arena, and third order effects, i.e., robots interacting with other robots, in order to correctly asses the consequences of actions. These third order effects are typical for swarm scenarios where large numbers of robots are loosely coupled by their interactions. This goes far beyond simple avoidance of the other robots but can more correctly be classified as strategic behavior.

12.2. Safety Definition

To be able to make decisions on what actions to use, a safety concept is needed. A very simple measure was chosen in order to make the decision process as transparent as possible to facilitate experimentation and debugging.

An action for a robot is considered safe if no other robot or obstacle is closer to it than some safety distance for any given time. This safety distance is chosen to be considerably

[1]For a detailed breakdown of the contributions of the different authors refer to section 1.5.

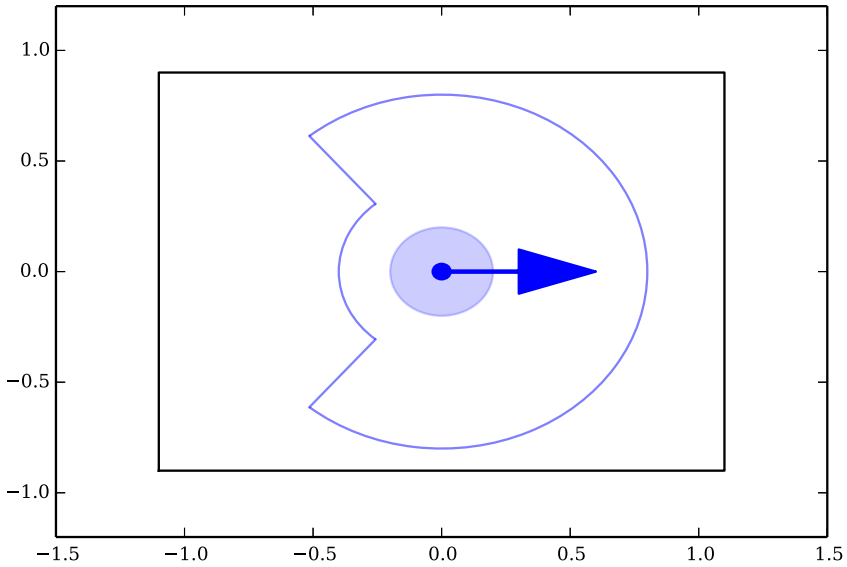


Figure 12.1.: The area of attention is indicated with a solid line. The lightly blue shaded area is the region defined by the safety radius.

larger than the range of the IR sensors of the robots.

In the context of the CE architecture this means that the CE has to check for every possible next action, so that no other robot gets closer than this safety distance using the predicted trajectory obtained by the simulation. The CE therefore literally evaluates the consequences of each possible action for this safety criterion.

Note that these predicted trajectories are not simple ballistic continuations of the current pose and speed of the other robots for a given action but also take into account what can be called second order effects, i.e., interactions of the robots with the environment, and third order effects, i.e., interactions between robots. In this regard the approach introduced here goes beyond most biologically inspired internal models considered in the literature, which are mostly body and physics models and do not take interaction of any kind into account.

12.3. Implementation Details

The architecture described in section 10.2 has been implemented as described in section 10.5. For the specific experiment conducted here however, some modifications to this implementation have been performed. In particular, some heuristics have been incorporated owing to practical necessities, for example the limited simulation budget (see section 10.4.4). In this section, the high level modifications are described. The particular implementation of the set of possible next actions and the corresponding safety values are described in section 12.4.2.

12.3.1. Attention Mechanism

The naive implementation loops the consequence engine through all possible actions. This is in principle the best possible option but not always feasible because of limited computational resources, i.e., a limited simulation budget (see section 10.4.4). Furthermore, the simulation error (see section 10.4.3) increases with time, so it might not make sense to simulate areas far away, which also implies long simulation times, since no significant information can be gained because of the large simulation error before an interaction actually occurs.

To address these issues, a simple heuristic in the form of an area of attention around the robot was implemented. Actions with goals outside of this attention area were not considered by the consequence engine. Furthermore, other robots outside of the attention area are not simulated to make more efficient use of the simulation budget. Coincidentally this also leads to a more realistic virtual sensing of other robots since it is similar to a limited sensing radius.

The shape and size of the attention area chosen is depicted in fig. 12.1. As a comparison, the size of the safety area used in the safety definition (see section 12.2) of the experiment is also shown and the physical size of the robot depicted is to scale.

The size of the attention radius was chosen purely based on empirical knowledge gained in simulation. The shape of the attention are was chosen to be similar to the ones often encountered in prey animals with a reduced radius at the opposite side of the direction of movement. This choice was purely empirical but seems justified since the robot is alway trying to move towards its goal so it is moving away from other robots in its back. The shape and size of the attention radius can be object to optimization, but performance was well within desired design criteria so this avenue of optimization was not further pursued.

12.3.2. Adaptive Simulation Time

The amount of time simulated into the future is a further parameter of the architecture. There is a minimum viable simulation time dictated mainly by physics in that the robot needs enough time to be able to react and avoid another robot. Furthermore, simulating too far into the future also poses a problem as it increases the probability of a dangerous situation. Simulating infinitely far into the future would lead almost certainly to a dangerous situation, resulting in an effectively paralyzed robot as all possible future actions lead to a dangerous situation. As the prediction error also increases with simulated time and the robot only has limited computational resources, these very long simulation times are not feasible in practice in any case.

A simple heuristic was designed to dynamically adapt the simulation time using the simulation results themselves. For that, a minimum and maximum simulation time (in this experiment $7.5\,\mathrm{s}$ and $15\,\mathrm{s}$) is fixed. The simulation times are adapted for each possible action individually while looping through the set of all possible actions. If a particular action does not lead to a dangerous situation, the simulation time is increased by 50% for the next update and if it leads to a dangerous situation, it is decreased by 20% and the simulation is re-run immediately. Both, increase and decrease of simulation times, are limited by the minimum and maximum simulation times. The asymmetry between increase and decrease is a balance between the limited simulation budget, the necessity to react to a dangerous situation as fast

as possible, and the desire to maximize performance of the overall algorithm by not choosing simulation times that are too short.

12.4. Evaluation Experiment

As described in section 12.1, one intelligent and five dumb robots were used for the experiment. The general experimental setup is described in section 10.5. For the experiment described here, part of the arena described in section 10.5.1 was partitioned off to form a more narrow corridor of $2.2\,\mathrm{m} \times 1\,\mathrm{m}$.

Units in this section are $[\mathrm{m}]$ and $[\mathrm{rad}]$ respectively if not stated otherwise. The origin of the coordinate system is placed in the center of the arena and the axes are parallel to the walls of the arena with the x-axis parallel to its long side and pointing towards to the goal of the robot.

12.4.1. Initial Conditions and Goal

The intelligent robot is placed in an arena of size $2.2\,\mathrm{m} \times 1\,\mathrm{m}$ at the initial position $(-1, 0)$ oriented towards the far end of the arena and is to proceed to the other end of the arena at $(1, 0)$ with a maximum speed of $0.1\,\mathrm{m\,s^{-1}}$. Furthermore, a safety radius of $0.22\,\mathrm{m}$ around the intelligent robot was chosen.

Five other robots are placed randomly in the area $[-0.5, 1.0] \times [-0.3, 0.3]$ with random orientations and a minimum distance to each other of $0.3\,\mathrm{m}$. They execute the simple action `GoStraight(v);Avoidance` with random but fixed speeds randomly drawn from the interval $v \in [0.6, 0.8]$.

As a baseline experiment the intelligent robot moves straight towards the goal using only its IR sensors for avoidance. Consequently this strategy leads to a high probability of another robot entering the safety radius even though actual collisions are avoided using the avoidance behavior. This naive baseline strategy is then compared to the one using the CE. To increase comparability between the two approaches, all experiments are initialized with the same initial conditions for both approaches. The initial conditions between pairs of approaches are randomized as described above.

Furthermore, the roles of the six physical robots in the experiments are chosen randomly to be either intelligent or dumb robots to mitigate inter-robot-heterogeneity between each pair of experiments (baseline and CE approach).

The experiments were also repeated purely in simulation. Thus in total, four types of experiments (baseline real, CE real, baseline simulation, CE simulation) were conducted, which were repeated 54 times (real) and 88 times (simulation) respectively.

12.4.2. Actions and Safety Values

Following the ideas developed in section 10.3.1, PFs are used for the basic safety values to describe the task. Specifically, the basic safety values are defined as

$$s(x, y) = 1 - \frac{10 - x}{30} - \frac{|y|}{300}, \tag{12.4.1}$$

This PF describes a trough with an incline from the start to the goal of the robot. This PF was the sampled on a 6×4 grid with dimensions of $2\,\mathrm{m} \times 0.8\,\mathrm{m}$ to create the set of next possible actions for the intelligent robot. Each of these actions was formed of the sub-actions `MoveTo(x,y)` and `Avoidance`. Even though this set of actions is discrete, the overall behavior of the robot is continuous and since the CE operated at about $2\,\mathrm{Hz}$, the robot can change the current action several times between moving from one grid point to another, effectively interpolating its actions between the grid points.

12.4.3. Results

One example run of the experiment is depicted in fig. 12.2. Shown are snapshots of the experiment with the predicted trajectories for both the intelligent robot and the other robots from the point of view of the intelligent robot. The CE loops through all possible action but for clarity, only the simulated trajectories for the best action, which was then selected by the AS, is depicted.

The reader is also referred to the recorded videos of the real experiments for a better illustration. What is remarkable for these experiments is how large the safety radius is in relation to the robot density of the arena. The robot has to follow a rather complex trajectory to avoid all other robots and safely reaches its goal.

To evaluate the effectiveness and efficiency of the CE approach, a statistical analysis of all experiments with real robots and in simulations was performed[2]. The four analyzed metrics were the time it took the robot to reach its goal, the distance covered while doing so, the fraction of time considered unsafe in relation to the complete run time and the number of simulations performed per time step as a cost measure. The results of this analysis are depicted in fig. 12.3.

The first thing to notice is how remarkably close simulation and real world experiments are. The two main reasons for this are first that the simulator itself is very accurate and well calibrated to the real world robots (see section 10.4.3) and second that there is almost no measurement noise in the system due to the high precision of the Vicon system used for the virtual sensing. In a real application, estimating the pose of other robots using for example a camera system is a difficult task.

As expected, the intelligent robot takes more time to reach its goal and covers more distance doing so. The increase is moderate; and is about 50% more time used and 30% more distance covered. These measures are not symmetrical since the robot can also stop and thus take longer to reach its goal without traveling further. Furthermore, variance in the results is increased for the intelligent robot, which is also to be expected.

The intelligent robot is a lot more safe, in simulation it is almost perfectly safe[3] but the improvement in the real experiments is still impressive. This improvement comes at the cost of having to do the simulations, while the dumb robot can act with a purely reactive

[2] See section 10.4.3 for the reality gap relevant for pure simulations.

[3] This is due to the fact that in pure simulation the possible simulated next actions are a direct continuation of the past trajectories, which were also simulated by the same simulator, as there is no reality gap (see section 10.4.3).

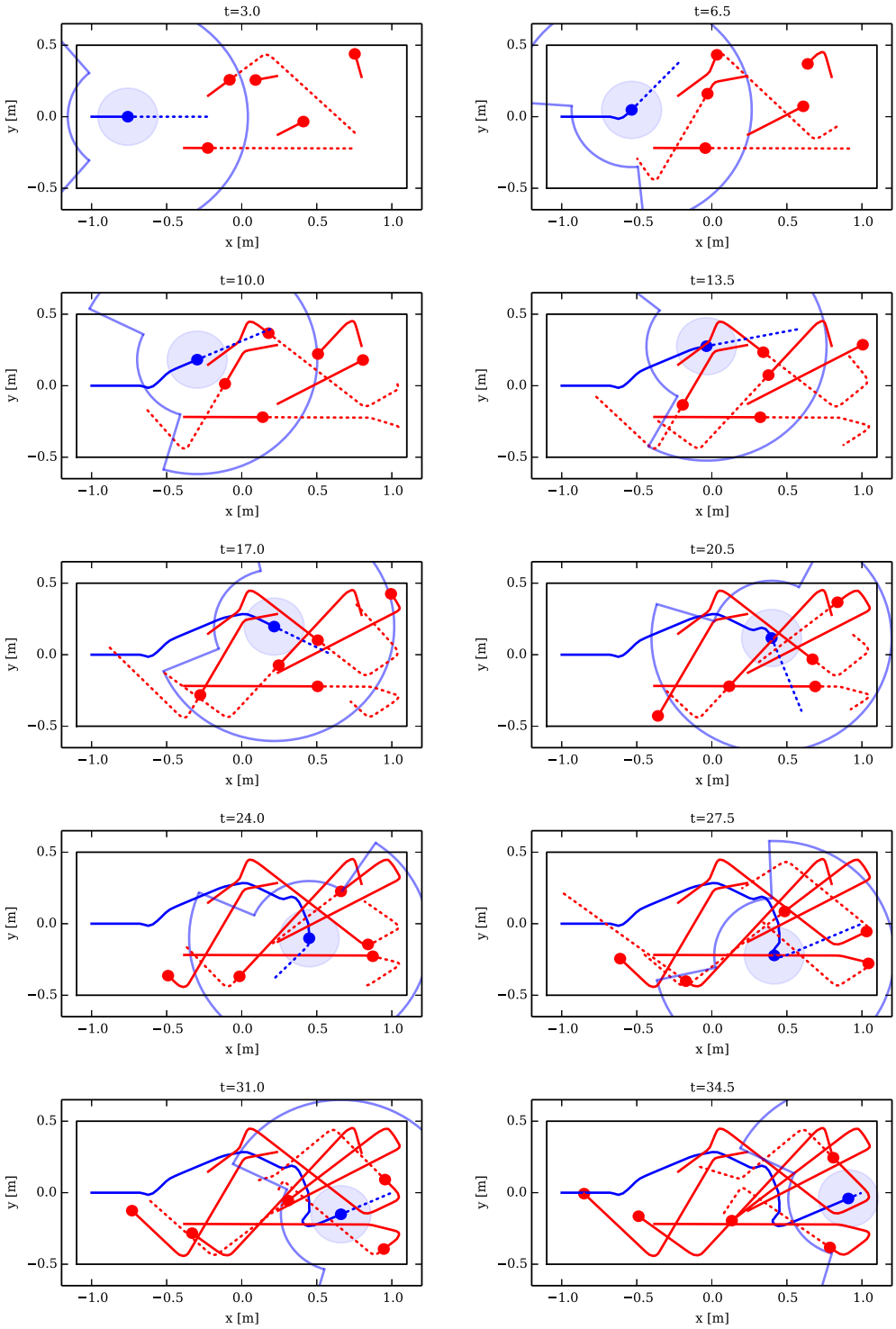


Figure 12.2.: Trajectories for one experiment with an intelligent robot in simulation. The intelligent robot is blue and the dumb ones red. Solid lines are real trajectories while dotted ones represent predicted trajectories. Only the simulation result for the best action is shown.

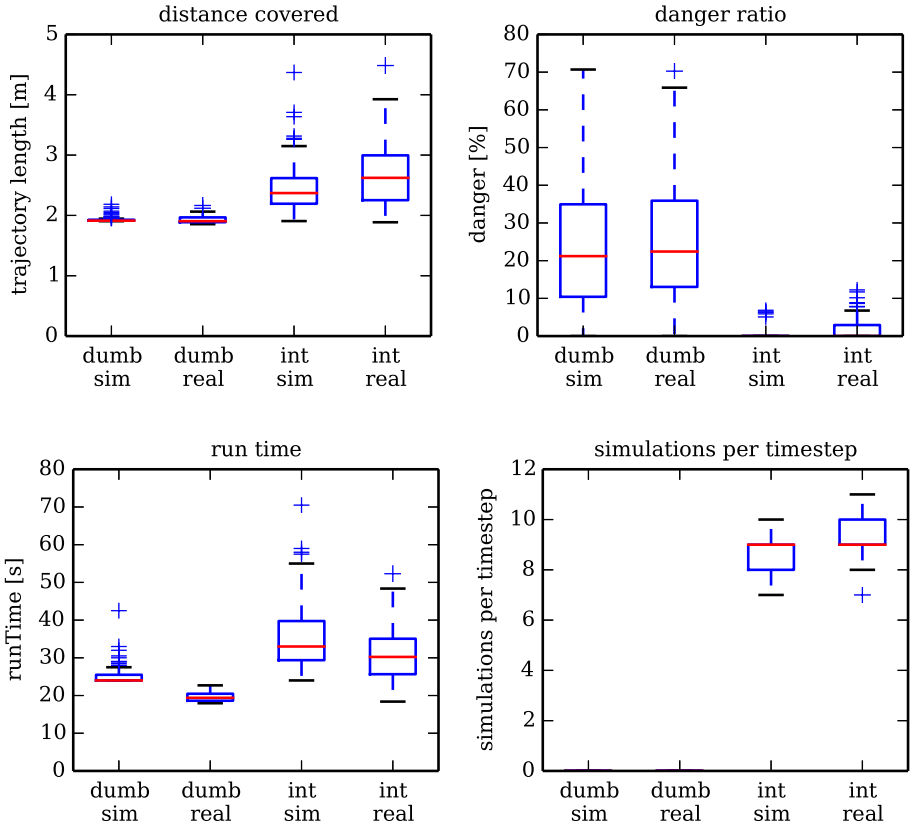


Figure 12.3.: Statistical analysis for 88 simulations and 54 real experiments considering four basic metrics and comparing them to the baseline experiment

controller. Here the trade-off between computational complexity and effectiveness between the two algorithms can be seen.

12.5. Discussion

Computing power of modern hardware is ever increasing, slowly allowing realistic embodied internal simulations to shift into the realm of possibility. This allows the paradigm of extensive internal simulation as opposed to purely reactive behavior to become more feasible. Nevertheless, the simulation budget has to be aggressively managed even on a modern computer to make real-time operation feasible. Using such an internal simulation based architecture like the CE, safety for mulit-robot systems can be increased even in situations where no direct communication between the robots exists. This is also a small step towards robots interaction safely with humans in natural environments such as an office corridor.

Furthermore, this experiment showed that the CE is able to simulate a considerable num-

ber of other robots of the order of magnitude of nearest neighbors in a typical swarm scenario and that it is able to correctly take into account not only robot-environment but also robot-robot interactions.

However, there are two important tasks that have been massively simplified in this experiment: sensing and learning of internal models. The sensing problem is solvable with modern hardware but also eats into the simulation budget as is is computationally intensive. Furthermore, questions such as object separation etc. are not fully solved as of this writing. Internal models have been learned in robotic applications. However, the ones demonstrated until now are on a very different complexity level than the full-blown internal simulator used in this experiment. It is yet unclear if and how this gap can be closed by purely learning from experience. In some situations where safety is so critical that a wrong choice can lead to fatal consequences, (one-shot) learning will be very challenging.

There are additional limitations of the internal simulation itself: the issue of the exploding search tree (see section 10.4.2) when doing simulations has not been solved yet and in this work a greedy depth-first search is employed. Furthermore, the controllers of the other robots used in the simulation are very simplistic and stateless. In principle, the methods used for internal self-modeling can be employed in these situations but have been only demonstrated in very simple scenarios until now.

12.6. Outlook and Future Work

There are basically four main possible avenues for potential future work. The most obvious avenue would be to use more complex robots instead of minimalistic e-puck robots to increase the number of free parameters and complexity to test the limits of the simulation tools and sensing in regard of the reality gap. Additionally, for actual embodied implementations, the sensing also has to be embodied so the OTL has to be implemented using the sensing modalities of the robot, for example a camera instead of an external tracking system. Furthermore, the a priori existent internal simulator as used in this experiment has to be replaced by learned internal models of the environment and the other robots/agents. And finally, the last big open question is how the challenge of the exploding search tree can be solved. One interesting option for that would be to use a particle filter style Markov chain Monte Carlo probability density estimation instead of using single discretized actions.

Chapter 13

Further Internal Simulation Experiments

13.1. Introduction

It has been shown that internal models can be used to recognize and select actions using internal simulations (Demiris and Khadhouri, 2006; Schillaci et al., 2012b,b). The models used in this context are usually learned and can only predict a single time step into the future. In this chapter, proof-of-concept experiments are presented, showing how the approach of using a full off-the-shelf simulator as used in the CE architecture (see chapter 10) can successfully be used for similar tasks. A first experiment shows the prediction error of the simulation can be used to recognize actions. A further experiment then shows a way to use the simulator to learn model parameters of the modeled controllers.

13.2. Prediction Error for Model Validation

13.2.1. Idea

This experiment uses the prediction error of the simulator run by a robot as a measure of the quality of the modeling, in this case mainly the modeled controller of a second robot. This measure can then be used to assess different controller models and therefore use the best one as the current hypothesis of what the second real robot is actually using as a controller. This implements a way of recognizing the actions of another agent.

Similar to the work by Millard et al. (2014a).

13.2.2. Setup and Experiment

The experimental setup consists of a $2\,\mathrm{m} \times 1\,\mathrm{m}$ arena and two e-puck robots with Linux extension boards. The infrastructure of the setup is the same as described in section 10.5. The Vicon tracking system is used as a virtual sensor to allow the robots to sense the position of the other robot.

One of the robots can use an internal simulation[1] and executes a controller with the sub-actions `GoStraight(0.8)` and `Avoidance` as described in section 10.5. This robot will be

[1] Practically, this means that it can access the simulation server run on an external computer as described in section 10.5.

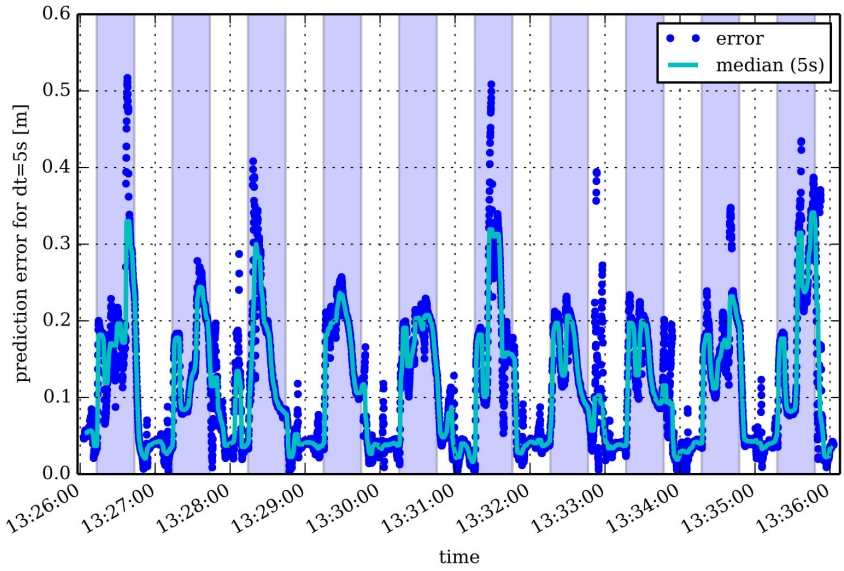


Figure 13.1.: Prediction error

called the leader from now on. The other robot, which will be called the follower, executes a controller with the sub-actions `Follow(distance)` and `Avoidance`. The sub-action `Follow(distance)` consists of a p-controller for the heading to point it towards the robot to be followed and another p-controller for the desired distance. This second p-controller has a significant higher gain for too small distances than for too large distances to help avoid collisions. This type of controller is known as a variable structure control (Emelyanov, 1970).

The follower switches the desired following distance to the leader every $30\,\mathrm{s}$ between $0.2\,\mathrm{m}$ and $0.4\,\mathrm{m}$. The leader tries to predict the behavior of the follower via its internal simulation but only uses a single modeled controller using a fixed desired distance of $0.2\,\mathrm{m}$. Thus, the predictions of the leader are wrong half of the time.

13.2.3. Results

The prediction error is defined in this experiment as the distance of the predicted position of the follower for a $5\,\mathrm{s}$ projection into the future to the real position of the follower measured $5\,\mathrm{s}$ after the moment at which the position of the follower was measured for the initial position of the simulation. Another option for the definition of the prediction error would be to use the mean squared error (MSE) of the complete simulated and measured trajectories. As the deviation of a simulation from reality is in general maximal at the end of a simulation period, the chosen definition is more sensitive albeit also more noisy (since the MSE acts as a low-pass filter). This prediction error is depicted in fig. 13.1 for the experiment.

The controller of the follower tries to keep it at a fixed distance to the leader so if the leader predicts the position of the follower with the wrong desired distance, this error should be

related to the wrong parameter. In general, this is only true for an open space, so there are some artifacts caused by the finite size of the arena. In practice these boundary effects lead to noise and outliers of the prediction error measurements. These effects are especially strong for cases when the leader avoids a wall and thus also interacts with the follower. This behavior can also be seen in the plot. For times when the model is wrong, the error is about $0.2\,\mathrm{m}$ higher than it is for times where the model is correct, reflecting exactly the $0.2\,\mathrm{m}$ error of the model parameter.

This experiment shows therefore that the prediction error of the internal simulation as a measure for the model quality and implement a mechanism to recognize different behaviors of other agents. This experiment is only a proof-of-concept and a real implementation of such an intention recognition algorithm would be much more complicated as it would have to deal with multiple internal models, i.e., controller models in this case.

13.3. Learning Internal Model Parameters

This experiment basically extends the previous experiment described in section 13.2. The prediction error of the internal simulation is used as a measure for the quality of the controller model and the parameters of this model are optimized to maximize this measure, thereby learning these parameters.

13.3.1. Idea

Models for controllers — in the context of internal models, controllers are also called inverse model — are usually learned directly on trajectories or as a coupled inverse-forward model pair, i.e., simultaneously with the forward model. Since the forward model is fixed in this experiment in the form of the internal simulator and the structure of the controller is therefore also fixed, the goal of this experiment was to show that parameters for a controller model can be learned using the prediction error of the simulator as a quality measure.

However, the prediction error is a compound of various error sources, internal ones, such as the calibration of the simulator to the real world, as well as external ones, such as a misalignment of the tracking system, i.e., measurement errors. This means that even a perfect controller model will lead to prediction errors. So in general this method might not be the most effective or efficient one for learning the controller model itself since the quality measure is not only a measure of the quality of the controller model but of the whole system of controller, simulator and measurements[2]. Nevertheless, this approach is feasible since in most applications the prediction accuracy is more important than the modeling itself.

[2]This means that if for example the simulator is mis-calibrated to the real robots by underestimating the speeds by 10%, the controller parameter for speed will be wrong by the same amount to compensate for the mis-calibrated simulator since that combination will minimize the prediction error. So the overall system performs optimally but the controller itself is not a correct representation of the real robot controller. Systematic topological errors however, such as a wrongly sized arena, will always lead to a non-zero error.

13.3.2. Setup and Experiment

The basic infrastructure setup again is the same as in section 10.5. For this experiment six e-puck robots were used in an arena of size $2.2\,\mathrm{m} \times 1.8\,\mathrm{m}$. All e-pucks were running a controller with sub-actions `GoStraight(speed)` and `Avoidance`. The constant speed parameter for the `GoStraight(speed)` sub-action was chosen randomly and independently for each robot from a uniform distribution in the interval $[0.5, 0.9]$ at the beginning of the experiment and was constant throughout the experiment. The experiment was run for around $10\,\mathrm{min}$ and the trajectories were logged.

After the experiment, the logged trajectories of the robots were used as initial conditions for simulations into the future and using the outcome of these simulations, prediction errors against the real measured trajectories could be calculated.

Since the learning of the model parameters constitutes an optimization, the quality measure, i.e., the prediction error, will be called fitness value from now on. Furthermore, this conforms to the nomenclature of the software library used to perform the optimization. The fitness value is calculated by using each trajectory sample as a set of initial conditions for the simulator, which then propagates these into the future by a fixed amount of time (in this experiment a simulation time of $5\,\mathrm{s}$ was chosen). The result of this simulation is then compared to the real trajectory following the initial conditions. The prediction error is then calculated as the sum of the MSE of the simulated trajectory against the real trajectories of all robots as the fitness value.

The python package hyperopt[3](Bergstra et al., 2013) was used with these fitness values as a fitness function and the speeds for the `GoStraight(speed)` sub-actions of the robot controllers as parameters. Hyperopt employs random search and Tree of Parzen Estimators (Bergstra et al., 2011) to sample the search space in an intelligent way. This approach was chosen because it can deal very well with awkward fitness-spaces and very limited fitness evaluation budgets. Furthermore, the speed parameters for the sub-actions of the controller models are indeed hyper-parameters of the model so it makes sense to use an algorithm specialized on hyper-parameter optimization[4].

The dataset generated by the experiment has around 6000 samples, which means for every evaluation of the fitness function, 6000 simulations have to be executed. The simulator can run at around 600 times real time, which means that a single evaluation would take $50\,\mathrm{s}$ plus the time to calculate the MSE of all simulation runs, which in total is a prohibitively long time. This problem is mitigated by using mini-batches for each fitness evaluation. For this 200 samples are drawn randomly without replacement from the trajectory samples and the fitness is calculated from these values only, which gives a speed-up of 30.

An alternative method would be to only use single samples from the experiment to calculate the fitness values. The nature of the experiment leads to two topologically distinct cases for trajectory samples: either all robots are just going straight and/or avoiding a wall, i.e., the fitness value is only dependent on the individual speeds but the speeds are independent

[3] `https://github.com/jaberg/hyperopt`

[4] Other optimization algorithms such as Broyden-Fletcher-Goldfarb-Shanno (BFGS) were also tried but didn't converge at all. This is not surprising since the fitness space is not very smooth and certainly not convex. Stochastic Gradient Descent worked but was a lot less efficient than hyperopt.

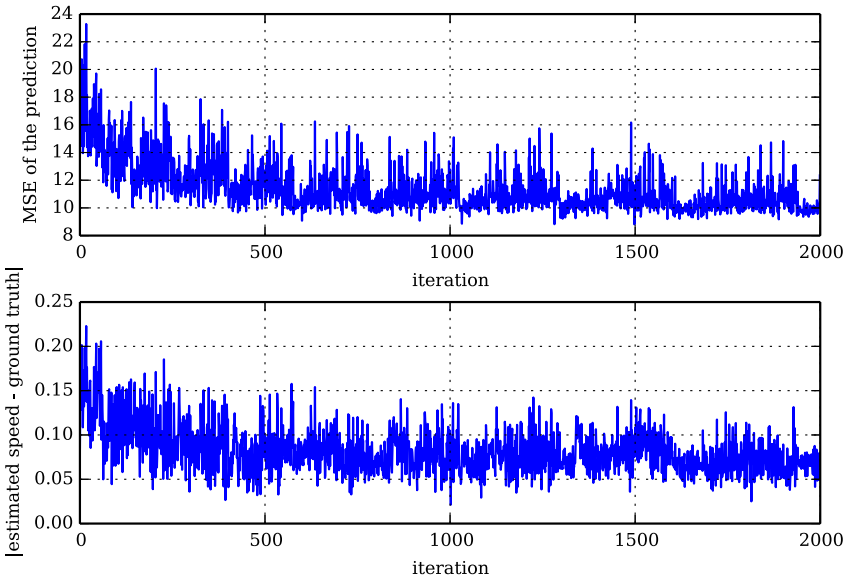


Figure 13.2.: History of the hyperopt optimization progress. The top plot shows the evolution of the loss value for hyperopt while the bottom plot shows the evolution of mean error of the actual speed estimates against the ground truth for all robots.

of each other. Or the robots are interacting directly with each other which means that the fitness value depends on the combination of the speeds of both robots, since some speed combinations might lead to the robots missing each other while other speed combinations lead to interaction. By using mini-batches these effects out are partially averaged out.

13.3.3. Results

The optimization process of the hyper-parameter optimization for the controller models of the simulation is depicted in fig. 13.2. The first plot shows the evolution of the fitness (hyperopt in fact uses a loss function) of the solution where every data point is calculated via a mini-batch. The second plot shows the actual error of the speed estimations against the ground truth (the recored randomly chosen speeds of the experiment).

The exact results of the optimization process found by hyperopt are shown in table 13.1. The mean error is in the range of 10%.

Note that neither the loss nor the actual error of the estimates converges to zero but to some non-zero constant. The reason for this behavior is the general reality gap of the simulator, which means that there are systematic errors of the simulation compared to real experiments. As shown in more detail in section 10.4.3, the main factor in this setup is that the tracking system is not perfectly aligned with the arena. In particular the arena is slightly rotated in relation to the tracking system, which means that heading measurements of the robots have a systematic error. Consequently, any simulated trajectory has an offset to the

Robot Name	Best Estimate	Ground Truth	Error
epuck22	0.763	0.721	0.042
epuck21	0.967	0.797	0.170
epuck12	0.743	0.710	0.033
epuck29	0.746	0.665	0.081
epuck32	0.807	0.744	0.063
epuck46	0.773	0.711	0.062
mean	0.800	0.725	0.075

Table 13.1.: Speed estimates for the robot controller generated by hyperopt with the prediction error as the fitness function

measured one, especially for straight segments, where the error grows linearly with the simulated time. In an infinite arena, this error would be of no consequence but as the area is finite, the error mainly shows up in boundary effects. Thus, the trajectory error as well as the error of the estimated speeds can never converge to zero.

This also shows a weakness of the approach in that it has to deal with errors external to the modeling itself, which are systematic to the simulator or measurements of the real trajectories. Nevertheless, the estimation of the model parameters work reasonably well for most applications. Even though the simulation has a systematic error offset, the optimization of the simulation error will find optimal solutions in the framework of this simulator, which is the best possible solution if the simulator itself is not to be changed by the algorithm.

13.3.4. Future Work

In this experiment hyper-parameters for an existing controller were optimized. The most obvious path of advancement would be to include more controller parameters into this optimization and ultimately learn the whole robot model, i.e., sensor, motor, and controller model, in such an optimization process. In principle this idea can be extended to the whole simulator itself, but getting to this point is a very ambitious task. The current state of the art are coupled inverse-forward model pairs learned from data for pure motor control for one robot being able to simulate one time step into the future.

Part IV

Conclusion

Chapter 14

Conclusion

14.1. Summary

This thesis started with the following problem statement:

> Investigate control strategies for autonomous mobile nodes of a wireless ad hoc network enabling them to work in a self-organized fashion, which only relies on local information.

This problem statement was then broken down into two main sets of issues related to the navigation of single robots in wireless networks and scalable self-organization of multiple robots in the context of network robotics.

Starting with the first set of issues, a literature review of the physical basis of wireless communication as well as technical basis of wireless networks was conducted. The results were presented in chapter 2. This showed that the spatial and temporal dynamics of wireless networks are complex in all but the most simple scenarios. Of special importance from a robotics point of view emerged the effect of small scale fading (see section 2.3.2).

To gain further insight into real dynamics of wireless networks, a series of measurements was conducted using real wireless networks in a regular office environment, sometimes also in the context of the Humboldt Wireless LAB (HWL). These measurements are presented in chapter 5 and formed the basis for further investigations. It allowed for developing an intuition for wireless networks, which proved to be an invaluable tool throughout this work.

Additionally, the literature review presented in chapter 4 showed that most of the challenges of wireless networks are already known and different solutions related to navigation in wireless networks have been proposed. A large number of localization algorithms for nodes in fully static nodes without mobility exists with or without the use of anchors, which are nodes with a priori known positions. For mobile nodes there are fewer algorithms to locate other nodes, most of which are either (pseudo) gradient-based or heuristical. Most of these algorithms however lack a theoretical background, especially in terms of convergence. There are also more complex algorithms using statistical models. Only a few of the algorithms have been implemented in real life and even less on actual robots (some have been tested carrying a laptop computer manually for example).

Exploring the outline and coverage area of a wireless (ad hoc) network is one of the basic tasks for a robot interacting with them. However, there existed no simple algorithm for exploring a wireless network for a single robot. Thus, a very simple and noise robust algorithm for this scenario was developed in chapter 6. It was evaluated with a focus on robustness to noise in simulation only because of technical reasons, which lead to an experimental large-scale evaluation being out of the scope of this thesis. The simple design of the algorithm proved to be very robust against large levels of noise.

Gradient based algorithms, for example source seeking, are popular in network robotics. However, there are very few theoretical results as to the convergence of these algorithms. A simple source seeking gradient based algorithms based on RSSI measurements was analyzed in chapter 7 using the mathematical framework of stochastic approximation by re-casting the algorithm into a commonly used gradient formulation in this framework. Textbook convergence criteria from this framework could then be interpreted in the context of RSSI measurements to show that the physical nature of these measurements and the network robotics context fit these convergence criteria if certain requirements for the algorithm are met. Furthermore, an experimental validation of these results was implemented on a ground-based robot using parameter values derived from the convergence criteria. Additionally, it became clear that this class of algorithms can also be used to solve more complex tasks than direct source seeking by constructing a more complex (virtual) measured variable instead of directly using measured RSSI values. For example, a new variable can be constructed combining RSSI measurements from two nodes in a way to solve the task of bridging these two nodes.

Building on the idea of formulating more complex tasks for network robotics in terms of measurable network parameters, an internal model based meta algorithm for navigation was developed in chapter 8. A more complex algorithm is needed for these kinds of tasks because convergence for gradient-based algorithms cannot always be guaranteed, for example due to local minima. The algorithms learns an internal model (see chapter 3) of the measured variable and uses this learned representation instead of using directly the measured quantity to chose the next movement step. This model is learned using an ϵ-greedy algorithm borrowed from reinforcement learning. It was implemented experimentally on a ground-based robot and evaluated in a real office environment performing the tasks of node finding as well as the bridging task suggested in chapter 7 to bridge two separated nodes. The algorithm has been able to solve both tasks successfully during working hours, i.e., in the worst case scenario from a measurement point of view with for example people walking around.

For large scale outdoor experiments, flying robots were investigated as a robotic base in appendix A. In particular, a large hexacopter with computational and sensing (regular sensing as well as measuring wireless network parameters) capabilities was developed and constructed as a flying mobile network node. Additionally, small flying copters were investigated for safe indoor experimentation. As these small copters cannot carry much payload, the sensing was performed externally using a motion capturing system. Unfortunately, experience with these flying systems showed that the technical and logistical challenges of using such systems was outside the scope of this thesis.

The second set of issues implied by the problem statement is concerned with self-organization of multiple robots in the context of network robotics. A literature review of algorithms in this context presented in chapter 9 showed that many algorithms have been investigated and developed dealing with issues ranging from node and network deployment over network optimization to network repair after faults. Further investigation however showed that there exist real testbeds only with small numbers of mobile nodes for these algorithms and no real world applications at all. The vulnerability of most swarm algorithms against partial faults of and/or malicious swarm member was identified as one of the main causes for this observation. Additionally, possible approaches from the literature to solve this issue were reviewed.

An internal model based architecture from the literature called Consequence Engine (CE) was identified as a very promising candidate to mitigate these real world challenges. The internal model is used to predict consequences of possible future actions, which are then evaluated using the predicted outcomes of the respective actions. The robot can then choose which of the possible future actions to execute by their respective evaluation.

As the CE uses an off-the-shelf internal simulator as the internal model, it is able not only to simulate simple actions of other robots but is, in contrast to most learned internal models, also able to simulate interactions between robots and the environment. However, more importantly it is also able to handle interactions between robots, which simple learned internal models cannot handle at the current state of the art. Thus it is able to also deal with swarm-like scenarios, which are in fact defined by the interactions between the swarm members, as encountered with algorithms for self-organization for multiple robots in the WSN context.

This architecture was subsequently implemented in chapter 10 on real robots. For this some minor modifications of the original architecture had to be performed to account for the implementation on real hardware. Using this experimental implementation, basic characteristics of the architecture were investigated. In particular the computational limits posed by real hardware proved to be a challenging problem leading to a limited simulation budget. Several heuristics to better manage this budget were proposed.

Using this implementation, several first demonstration experiments were implemented in chapter 11 to test the ability of the CE to make a robot cognizant of the consequences of its own actions on itself and others. These experiments consisted of an arena with a virtual hole, the robot running the CE and one or two other „naive“ robots. These experiments showed that the CE enables the robot to not only avoid the hole successfully, by being cognizant of some paths leading to falling into the hole, but also that it enabled the robot to rescue other robots from falling into the hole by blocking their path. This showed that the CE indeed makes the robot cognizant of the consequences of its action on itself as well as on other robots. Curiously, also a dilemma situation could easily be constructed in which the robot could not rescue both other robots at the same time leading to the robot often not rescuing any of them because it could not decide clearly which of the two to rescue.

In order to test the capabilities of the CE to simulate a larger number of other robots, specifically in a situation involving interaction between several robots, and to show how it can be used in a safety context, a further experiment was conducted in chapter 12. A robot using the CE had to move through a narrow corridor avoiding five other robots moving

around randomly going straight and performing obstacle avoidance on the walls as well as on each other. While moving though the corridor, the robot had to move in a way that no other robot got closer than a given safety distance. This is a exemplary task for a robot moving through a human environment like an office while avoiding physical contact with humans. This experiment showed that the CE is not only able to deal with numbers of other robots of the order of nearest neighbors in a typical swarm scenario and is able to take into account complex robot-robot interactions, and also that it can be used to implement practically relevant safety scenarios.

Furthermore, several proof-of-concept experiments related to fault detection as well as model learning and validation were performed in chapter 13. A first experiment showed how the prediction error of the internal model used by the CE can be used to validate the model against real measured behavior of another robot. This can be used to validate models of other robots, and it can also be used to check for faults in other robots against known or expected models. A second experiment showed how model parameters can be learned from observed behavior purely using the prediction error of the simulation as a metric. However, both experiments are only at a proof-of-concept stage since such an approach is well known from the literature and the experiments only validated that the CE is in principle capable of performing these tasks.

In summary, both aspects of the problem statement have been addressed. First, in order to gain understanding of real world dynamics of wireless network parameters, exemplary measurements of network parameters like RSSI were conducted. For single robots, a number of algorithms, mainly related to navigation in wireless networks were developed, implemented and evaluated experimentally mainly using ground-based robots. Flying robots were also investigated as a robotic platform but proved to be technically too challenging. First a robust and simple algorithm for network exploration was investigated. Next two algorithms for source seeking, based on gradients and internal models respectively, were investigated. For the gradient-based algorithm, analytical convergence criteria could be given. Furthermore, it was shown how these algorithms can also be applied to more complex tasks. Algorithms for self-organization with multiple robots in the WSN context were found to have been investigated extensively in past literature. However, there are only very few experimental implementations and no real world ones. Reliability of swarm systems against partial faults and malicious agents has been identified as one major cause. To solve that, an internal model based architecture has been implemented from the literature on real robots. Several experiments have been conducted to demonstrate the capabilities and properties of this architecture. These experiments have shown that the architecture enables a robot to be cognizant of the consequences of its own actions not only on itself but also on other robots. It can also deal with a number of other robots of the order of nearest neighbors in typical swarm scenarios as well as more importantly with robot-robot interactions. Furthermore, proof-of-concept experiments have shown that in principle the internal model used by the architecture can be learned and either validate the robot models against real behavior or validate robot behavior against known/expected models as is needed for fault detection.

14.2. Discussion

Wireless networks, in particular measured RSSI values, have proved to be an excellent surrogate in experiments for other scalar fields often encountered in robotic research like for example chemical concentrations in chemotaxis. Unlike, for example, gas clouds for chemical concentrations, they are easy and quick to setup in non-specialized experimentation environments such as in a regular office. There are other common replacements, like luminosity fields or purely simulated fields, but wireless networks can, unlike luminosity fields, easily extend through walls and around corners and also, unlike purely simulated fields, are not dependent on exact robot positions, which usually need some sort of tracking system. Additionally, they can be used for experiments ranging from several meters to tens of meters in size. However, and that can be seen as an advantage or disadvantage, depending on the experiment, they show more complex dynamics as discussed in chapter 2.

The algorithms related to network robotics for single robots presented here have been shown to perform well even in indoor scenarios. This is especially impressive since indoor scenarios are prone to create complex RSSI landscapes due to a large number of potential scatterers. On the other hand, indoor scenarios such as office environments often also lead to temporally complex RSSI measurements because of non-stationary scatterers, for example people walking around. Furthermore, these environments also often show elevated levels external interference due to the large number of different wireless networks and technologies used in these areas ranging from other IEEE 802.11 networks over bluetooth devices or wireless keyboards to microwave ovens. The main insight leading to this performance was to specifically take into account the effects of small scale fading and consequently to find ways of mitigation, as most algorithms are not robust against them [1] Furthermore, anticipating these challenges allows for their treatment starting at initial conceptual phases during the development of algorithms.

In contrast to the other network robotics algorithms presented here, the exploration algorithm has not been experimentally implemented yet. As this algorithm deals with large scale wireless networks consisting of multiple network nodes, a robust outdoor robot base would be needed to perform an experimental validation of the algorithm. For this, a flying robot would be ideal because, in contrast to a ground-based robot, it would be able to execute the algorithm more closely to the simulation since it would have to deal less with obstacle avoidance and would be able to operate in almost line-of-sight conditions in relation to the network nodes. However, even though a flying robot designed specifically for network robotics was manufactured and implemented as a prototype and several initial experiments were performed, it never reached operational status due to technical and logistical issues which proved to be out of the scope of this thesis. Nevertheless, given an adequate flying robotic platform an experimental validation would be worthwhile.

Contrary to originally anticipated, no actual algorithms for self-organization for multiple robots in the context of network robotics were investigated. A literature review of this area showed that even though a large number of different algorithms dealing with a variety of scenarios and tasks has been developed in the past and published in the literature, almost all

[1] See chapter 7 for the discussion of small scale fading and its impact on convergencei.

of them have only been simulated and never evaluated experimentally. Further investigation showed that typical experimental testbeds related to network robotics with multiple robots, i.e., mobile network nodes, consist only of small numbers of robots. Furthermore, there are no real world application of these algorithms to date. The fact that swarm systems, and most systems consisting of mobile network nodes can be counted as swarm systems, are not robust against partial failures and/or malicious nodes has been identified as one of the main challenges for real world experiments and applications. Therefore, an architecture from the literature able to mitigate these issues — the CE architecture — was implemented and evaluated experimentally instead of directly working on algorithms for self-organization in WSNs. These experiments have shown that the CE can indeed in principle be used to solve the identified problems. However the results of these investigations have not been directly applied to the special case of algorithms for self-organization in the context of network robotics yet. Consequently, more work has to be done if the two areas, network robotics for single robots as discussed in part II, and the results from working on the CE architecture in part III, are to be integrated to finally enable real-world experiments for this special case of self-organization in the context of network robotics. Only real world experimental evaluations can make existing algorithms more applicable to real wireless networks, which is why the work on the CE architecture was crucial.

The implementation of the CE architecture from the literature has been shown to perform remarkably well in experiments. It is able to deal with a wide range of situations related to the internal simulation of other robots, interactions between other robots and also hypothetical reactions of others on possible actions of the robot running the CE. These are the basic building blocks to make sense of situations dealing with more than one robot. Starting from this implementation and using the knowledge about its properties gained from the basic evaluation experiments presented here, there are a large number of options on how to continue (see section 14.3). Furthermore, this also shows how useful and versatile the paradigm of internal models is as it can be employed in a broad range of areas ranging from basic motor control to minimally ethical robots.

In this context it is worth mentioning that one of the biggest improvements on the abstract architecture of the CE was to actually automate experimentation. This is especially important in the area of swarm like multi-robot scenarios, i.e., ones with a robot running the CE and several other robots (the maximum total number of robots used in experiments in this work was six). In practice this includes bringing the robots into initial conditions for the experiment, monitoring the individual robots for errors, and coherent data collection for later analysis of the experiment which has to be able to deal for example with different clocks of the robots and asynchronous data collection as the robots are operating independently of each other and the CE and concurrently.

However, there is a gap between current learned internal models, which are most of the time body-centric, and internal simulation as used by the CE, which also includes others and the environment. In the implementation of the CE, the internal simulation existed a priori in the form of a powerful off-the-shelf robot simulator and was not learned form experience. Furthermore, the robot controllers of the other robots were also known a priori. However, some work has been performed showing how the controllers can in principle be learned and validated using the internal simulation mechanism. Nevertheless, it is not at all obvious how

the gap between learned forward models as used, for example, for motor control, and the off-the-shelf internal simulator is to be closed. Additionally, the way the CE was implemented employed a lot of shortcuts, for example using a tracking system as a virtual sensor.

14.3. Future Work

For clarity, possible future work is split into parts following the structure of this thesis. As the discussion of the work in section 14.2 has shown that additional future work is necessary to integrate both parts of the thesis together. Consequently, necessary steps to achieve this are included here as an additional third part.

14.3.1. Network Robotics with Single Robots

There are several technical improvements to be made. As already mentioned before, directional antennas can improve navigation as they enable directional measurements instead of purely scalar ones. This could improve gradient estimations and on the other hand could also lead to new algorithms not relying on gradients at all. However, most directional antennas do not have a very distinct directionality in that they either cover a broad range of bearings or have side-lobes (see section 2.2.3), so their use needs some consideration. Furthermore, directional antennas are often relatively big and heavy so they cannot be used on all robots depending on their size. This is especially important, for example in flying robots or small robots such as e-pucks.

As also already mentioned in section 14.2, the algorithm for network exploration was not implemented experimentally and would ideally be implemented on a flying robot because it is designed for large scale operation. In general, large scale and outdoor scenarios would be interesting evaluation scenarios for all network robotic algorithms presented here as these scenarios are similar but distinctly different to indoor ones. Furthermore, there are only very few examples dealing with 3D characteristics of wireless networks in the literature, which could be investigated with flying robots.

What has also not been tested extensively in this work is to use a mobile node in a live WSN situation, i.e., one with real network load. In all experiments, only dummy packets were generated at fixed rates. In a situation with network load, the network packets generated by the actual network load could be used for the RSSI measurements instead. This means that no additional artificial network load has to be generated for navigation, but also that RSSI measurements become asynchronous, which has to be accounted for in the algorithms.

The algorithm presented in chapter 8 used a Simultaneous Localization and Mapping (SLAM) algorithm based on a line reading from a Kinect sensor for position estimation. Figure 8.2 shows a RSSI map overlaid with the corresponding map created by the SLAM algorithm created during this experiment. Ambiguities in the SLAM algorithm when matching these sensor readings to the map could be mitigated by integrating RSSI measurements directly into the sensor readings for the SLAM algorithm.

Cross-layer optimization is another promising direction for further research. In chapter 7 and chapter 8 first steps to formulate more complex goals in terms of RSSI measurements

such as bridging two nodes were performed. In principle this approach can be extended to other network parameters such as for example PDR. However, these measurements are often more difficult to obtain as the information is not contained in a single packet the way RSSI is but often has to be calculated from a series of packets as most of these parameters are rates. This means that measuring these parameters constrains the movement of the robot to some degree as the complete measurement has to be performed ideally at the same point in space.

All algorithms related to network robotics for single robots presented here assume that all other nodes are stationary. This naturally leads to the question of if and how these algorithms perform if other nodes are also mobile. Following this, are queries related to self-organization, and also related to modifying the presented algorithms to account for node mobility.

14.3.2. Self-Organization of Multiple Robots

The simulator used as the core of the CE architecture can be improved in a number of ways. At the moment the simulations are performed in a purely greedy fashion (see section 10.4.2), which is the most simple but not the optimal strategy for traversing the search tree. Furthermore, the actions are at the moment discretely sampled on a grid (see section 10.3.1) instead of using actions from a continuous space as would be natural. Additionally, measurement noise is not accounted for in the simulator, which is not as critical when using a motion capturing system as a virtual sensor but becomes an issue as soon as internal sensing for example with a camera is used.

All these issues could be solved at once by using a particle filter style Markov chain Monte Carlo (MCMC) (Andrieu et al., 2010) type simulation where the future simulation steps are described as a Markov chain which is traversed by particles. These particles could then incorporate measurement noise as a priori distributions and account for future decisions of the robot during the time projected into the future by simulation. After performing several simulations, for which the sampling has to be performed conforming to the requirements of MCMC, they form a posterior distributions of future outcomes, i.e., robot trajectories for the robot running the CE and all other robots. This then integrates all possible future actions, decisions of the robot during the projected simulation time, and measurement errors. The set of possible future actions is then replaced by a Markov chain description. In contrast to the current implementation of the CE however, this procedure is a lot more computationally intensive as MCMC sampling is very computationally costly. Nevertheless, the individual simulation runs are completely independent of each other and thus embarrassingly parallel. Therefore, the complete architecture could be scaled up by replacing the single simulation server (see section 10.4.1) with several identical copies running on different computers and a transparent load balancer in front of them distributing incoming simulation requests. More advanced attention models based, for example, on the potential dangerousness of other robots could help to mitigate these increased computational costs to some degree.

At the moment the heavy lifting of the sensory input for the CE is performed using a motion capturing system with special markers on all robots. This obviously has to be replaced by for example a camera system for embodied sensing in a fully autonomous robot. However, this sensing problem is in general still not fully solved.

The next step in terms of robotic implementation will also be to use more complex robots for example a humanoid robot instead of the very simple wheeled e-puck robot used here. This then implies a more complex, i.e., computationally expensive, simulation and much more complex controllers for the robot. This will make the whole system more complex and is therefore a very good test for the general architecture and solving these problems will also lead to new insights into how to design such an architecture in the most optimal way.

In the current implementation of the CE the controllers of other robots are known a priori. In principle, as the experiments in chapter 13 have shown, these controllers can also be learned in the context of the CE using the internal simulator. These experiments however are only proof-of-concept experiments, which only learned controller parameters and not complete controllers, and can be vastly improved. In principle this could also include forward, i.e., in this context body kinematic models, of other robots. This also means that the CE could in theory constantly check its predictions against measured behavior and thus validate and/or correct its internal models used by the simulator.

The constant validation and correction of the internal models is directly related to fault detection and/or detection of malicious robots as the fact that the models show high prediction errors can also mean that the robot is showing unexpected behavior. This was also already shown in principle in chapter 13 as a proof-of-concept but more work needs to be done to reliably perform fault detection. One major conceptual difficulty for this will be to decide between a wrong internal model and unexpected behavior.

A further conceptual question will be how two robots, both running a CE, will resolve the problem of recursion in the simulation similar to the famous Prisoner's dilemma. This dilemma occurs as one robot tries to simulate another robot also simulating the first robot simulating itself and so on. There exists research in psychology dealing with the question of how humans solve that dilemma, which could be a starting point for experiments. However, as the daily experience of trying not to collide with other pedestrians when trying to avoid each other on the sidewalk shows this is not a trivial problem.

14.3.3. Assembling the Pieces

The initial goal of this thesis was to create algorithms for a WSN consisting partially or completely of mobile nodes. As discussed, self-organizing swarm like algorithms lead to issues for real implementations related to partial faults and/or malicious nodes. To solve this, the CE architecture from the literature was identified, implemented and evaluated. However, this architecture has not been applied to the special case of WSN yet.

One way to achieve this would be to directly include the wireless network into the internal simulator, either as a full network simulator or as a physics simulator only calculating signal strength fields. That way algorithms for self-organization in WSN could be directly simulated by the CE and could therefore implement for example fault detection. Without these additional simulation capabilities, the corresponding algorithms cannot be easily simulated as sensory input for the robot controllers would be missing. This wireless network simulator could also be, for example, learned in the style of the internal model used in chapter 8, which would mean to mitigate large part of the computational complexity of a full wireless

network simulator. However, in principle, off-the-shelf wireless network simulators could be readily included into the internal simulator of the CE.

Besides fault detection, the detection of malicious nodes is a critical problem in WSNs in general, and also for static networks. A mechanism similar to the CE presented here could be used to detect unexpected behavior as in usual WSN applications, the algorithms run by the individual nodes are known a priori, which eliminates the ambiguity between false models and unexpected behavior.

In addition, the CE could then also be used to optimize network parameters, for example by trying out different parameter configurations of the WSN algorithms.

Furthermore, results presented in part II on algorithms for network robotics with single robots can be readily transferred to multi-robot scenarios. In particular, the convergence results for gradient-based algorithms dealing with, for example, small scale fading can be directly transferred and are often not taken into account in WSN algorithms in the literature[2].

[2]This is probably due to the fact that the algorithms are almost never evaluated experimentally, so these kinds of issues are never encountered. Almost all simulators used in this context only use effective physics models, which cannot produce effects such as small scale fading and approximate it with a noise term, which however is, in contrast to small scale fading, not spatially deterministic (also see the discussion in section 2.1.3).

Part V

Appendix

Appendix A

A Flying Robot for Network Robotics

A.1. Introduction

This work was partly presented in (Blum and Hafner, 2012)[1].

Flying robots are a very attractive choice as mobile robot bases for network robotics since they are less constrained in their movement than ground-based robots, especially for large scale experiments and applications. As discussed in section 9.4 there are already some experimental testbeds consisting entirely of flying robots. Additionally to being more flexible, wireless dynamics for flying robots are often less complex as compared to ground-based ones since they often operate in line-of-sight conditions well away from potential scatterers and sources of interference.

Using full-scale flying robots interacting with real wireless networks enables the robot to work with the real network dynamics, which have a fixed spatial scale defined by physics. On the other hand, those robots are rather complex, which means that logistics and maintenance can pose its own challenges. Furthermore, because of safety and legal issues, one human pilot is needed on standby for every flying robot to take over control manually in case something goes wrong. This means they are best used in small numbers.

A.2. Environment

IEEE 802.11 WLAN is used as a wireless technology. This technology uses unlicensed ISM bands (2.4 GHz and 5 GHz), which makes it suitable for experimentation because — in contrast to for example cellular networks — protocols, radio parameters etc. can be varied in a wide range. For a more detailed discussion of this issue refer to section 2.2.2.

As a testbed the HWL (Zubow and Sombrutzki, 2011) is used, which is a large-scale wireless mesh network installed on the campus (Campus Adlershof Humboldt-Universität zu Berlin). It consists of about 100 indoor and outdoor nodes. The indoor nodes are placed in several buildings, forming a fully connected wireless network. The outdoor network covers almost the whole campus enabling extensive outdoor experiments. The outdoor network can be combined with the indoor network to improve connectivity between the buildings.

[1] For a detailed breakdown of the contributions of the different authors refer to section 1.5.

Figure A.1.: ArduCopter base system hexacopter

A.3. Platform

For the realization of a flying robot a multicopter was chosen because it enables complete 6 degrees of freedom (DOF) navigation while still being self-stable due to its internal sensors and basic control unit.

A multicopter usually consists of an even number of rotors rotating in clockwise and counter-clockwise direction to cancel the torque of the rotating rotors, an internal measurement unit (IMU) with 6 DOF and a microcontroller for the basic control loops keeping the robot stable. Additionally, a pressure sensor and an ultrasonic range finder are often used for altitude control.

The base is an ArduCopter[2], which is a community-based open source multicopter platform. Six rotors were chosen as a compromise between lift, size and weight. It is depicted in fig. A.1. It weights about 1.2 kg and can lift about 1 kg of payload. The ArduPilot Mega, which is the Arduino based microcontroller board, can be addressed using the MAVLink Micro Air Vehicle Communication Protocol by the PIXHAWK Project [3]. The robot has, in addition to the IMU, a pressure sensor, an ultrasonic range finder, a compass and a GPS module for reference position measurements during navigation experiments.

In autonomous mode, an Intel Atom based board[4] running Linux exchanges control messages with the microcontroller, implementing the algorithm under investigation. The algorithms can be implemented in the context of the robotics framework MAVHUB[5], which is a flexible and lean middleware, abstracting from the actual hardware and interfaces using

[2] DIY Drones, the ArduPilotMega project `http://www.diydrones.com`
[3] `http://www.qgroundcontrol.org/mavlink/start`
[4] COMMELL LP-170C `http://www.commell.com.tw/`
[5] `https://github.com/calihem/mavhub`

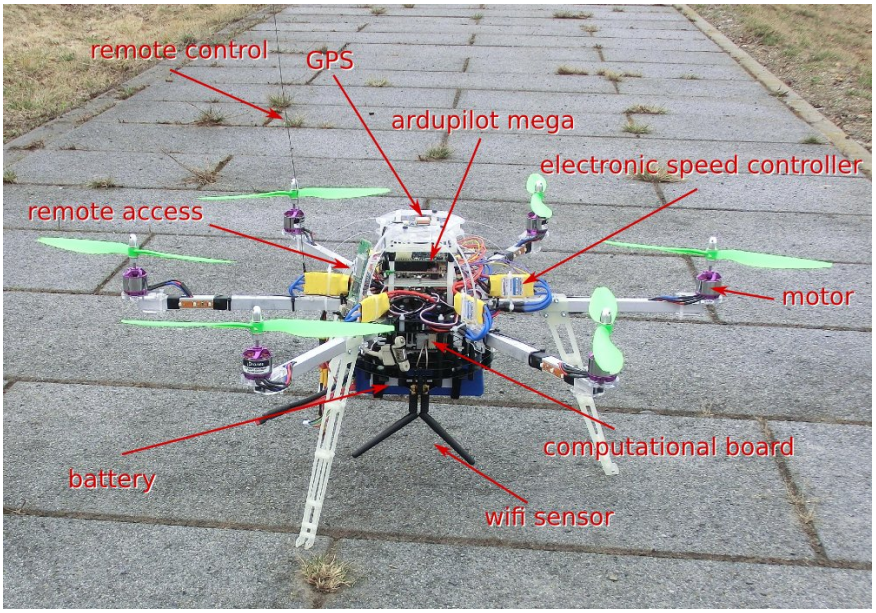


Figure A.2.: ArduCopter hexacopter with network robotics hardware

MAVLink messages as a backend. For WLAN connectivity, an Atheros AR9280 based dual band mini-PCIe card is used. The complete setup is depicted in fig. A.2.

For control and logging purposes, the WLAN can in principle be used. However, this additional traffic can in theory interfere with measurements and is also dependent on the Linux system. In consequence, traditional 35 MHz control for the underlying base was also implemented, as a safety measure as well as to not interfere with measurements, and is able to override the control commands issued by the Linux system. This control architecture of the robot is illustrated in fig. A.3.

In addition to its own WLAN capabilities, it is planned to incorporate a complete HWL indoor mesh node into the robot. For this, the electronics will be stripped from its case and mounted on the robot. This facilitates the use of all software developed in the context of the HWL testbed and helps making all measurements repeatable without any bias due to hardware differences in respect to stationary nodes.

The Click Modular Router (Kohler et al., 2000) is used as a basis for all WLAN measurements. In addition to its known strengths — scalability, speed and modular design — it is the base for the software used in the context of the HWL testbed. It is therefore very convenient to also use it as a userspace application on the computational board. This allows for a very detailed analysis of network traffic and access to a lot of hardware features of the AR9280. If the full feature set of the HWL testbed is not needed, simply python scripts and tcpdump/libpcap[6] can be employed as a prototyping alternative.

[6] http://www.tcpdump.org/

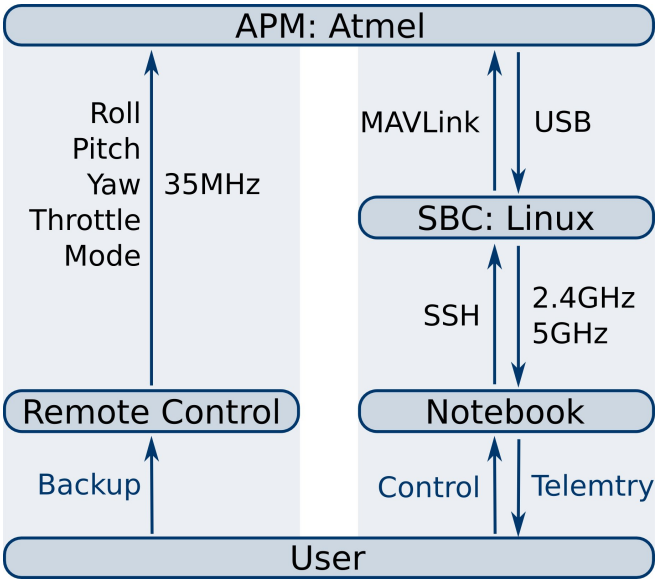


Figure A.3.: Control architecture for hexacopter

A.4. Experiments and Algorithms

Since sensory inputs are not static, but are gained through a closed sensorimotor loop and have different dynamics, concepts of sensorimotor interaction have to be applied to this new domain. Sensorimotor interaction (Pfeifer et al., 2007) is therefore one of the most important concepts for the design of algorithms since sensory inputs, here in the form of network and wireless parameters, are shaped by the motor interaction between the world and the robot movement. The algorithms then exploit the thereby generated correlations in the sensory inputs and the correlations between the sensory inputs and the motor commands.

Navigation is one of the key skills for an autonomous robot. Unfortunately, the information which is encoded in the network and wireless parameters is not directional at a specific point in space. Bacteria face the same problem when trying to find food (Berg and Brown, 1972) because they are too small to be able to detect effective gradients in concentration. During evolution they developed ways to engage in chemotaxis despite this limitation. They continuously switch between a random tumble and moving in a straight line where the duration of the movement in a straight line is governed by the amount of integrated gradient experienced while moving on the straight line. Interestingly, this is exactly exploiting the sensorimotor interaction. Algorithms developed in this spirit, which could be implemented on the flying robot, are presented in part II.

Further inspiration can be gained by the study of vision algorithms. Warping (Möller, 2009) for example, which is a local visual homing algorithm can enable the robot to find its way back to its home location. To achieve this goal the knowledge of the exact home location is not necessary since warping yields a vector pointing in the home direction. Following this vector, the home location can be reached. Adapting these ideas can facilitate the development of navigation algorithms based on for example the measured SNIR of WLAN base stations.

A.5. Discussion

Flying robots proved to be technically challenging. They required meticulous maintenance as faults of single robot components can result in disastrous faults for the complete robot. Software faults can often be overridden by manual control but need a skilled human pilot to bring them back into a stable flight regime. These reasons mean that a flying robot is prone to being destroyed by faults resulting in time and resource intensive repairs.

Finally, the legal situation in Germany requires one human pilot on stand-by for every flying robot to be able to react in case of an emergency. This makes experiments with more than one or two robots virtually impossible or at least logistically challenging.

Thus, flying robots as an experimental platform were unfortunately outside of the scope of this thesis.

Appendix B

Flying Mini Copter in a Tracking System

This work was presented in (Blum et al., 2014)[1].

B.1. Introduction

Small-scale model systems in contrast are a lot more safer to use and easier to maintain. Because of their small scale however, they cannot be used for experiments dealing directly with the physical characteristics of wireless networks. Furthermore, they cannot be used outdoors except for very windless days. Their main use is as a testbed to study the interaction between several flying nodes in a three-dimensional space.

B.2. Motivation

Providing an experimentation platform for the interaction between humans and flying robots is a challenge from a safety point of view because of rapidly spinning rotors. One approach is that followed by Lee et al. (2013); Pitman and Cummings (2012) providing a safety layer between the untrained operator and the robot. Intrinsic safety in particular is a major issue with flying robots due to potential crashes and contact with propellers. An approach for reducing danger of self-destruction for the robot itself is described by Klaptocz et al. (2013); Briod et al. (2013a). One general way of reducing the potential for harm is to reduce the size and mass of such robots. Here a system using very lightweight ($\mathcal{O}(50\,\mathrm{g})$) quadrotor helicopters is presented which can be used for indoor interaction experiments.

B.3. Approach

While it is currently quite challenging to realize full onboard sensing and behavior generation (control) of very small quadrotors (Briod et al., 2013b), results from the proposed experiments can easily be transferred from the tracking system case to the full onboard case because these results relate to general features of robot behavior.

[1] For a detailed breakdown of the contributions of the different authors refer to section 1.5.

Figure B.1.: Modified Walkera Ladybird with markers for the tracking system

As discussed above, a small quadrotor, the popular toy Walkera Ladybird [2], was chosen. The manufacturer designates it to be safe for children from the age of 14. It weighs only $31\,\mathrm{g}$ (with markers) and has a diameter of approx. $15\,\mathrm{cm}$ making it inherently safe even when accidentally hitting the human. Five reflective markers were added as can be seen in fig. B.1 to enable 6D tracking. The modified quadrotor is depicted in fig. B.1. In addition to human safety, the weight and size of the robot also make it almost invulnerable to crashes etc.

B.3.1. Control

Controlling a small flying robot in the direct vicinity of a human and facilitating interaction requires a closed-loop system in order to enable intuitive and safe behavior of the robot. Weight constraints on a small robot are severe, thus external control and sensors were used.

The copter is tracked using an external tracking system consisting of six Natural Point OptiTrack Flex13 Cameras[3] yielding 6D positions of the robot at 120Hz with an accuracy of about $0.5\,\mathrm{mm}$ and a latency of less than $10\,\mathrm{ms}$. These positions are streamed over ethernet to the computer controlling the robot using a Natural Point protocol.

The robot is controlled using the MagicCube remote by Walkera which is controlled by an Arduino running an interface program[4], which is attached to the control computer. This setup is depicted in fig. B.2. The computer receives 6D positions from the tracking system and sends motor commands to the robot. The Mavhub[5] infrastructure is used to deal with the control in quasi-real-time. The robot itself has onboard control loops for its attitude based

[2] `http://www.walkera.com/`
[3] `https://www.optitrack.com/`
[4] `https://github.com/azz2k/mCube`
[5] `https://github.com/calihem/mavhub`

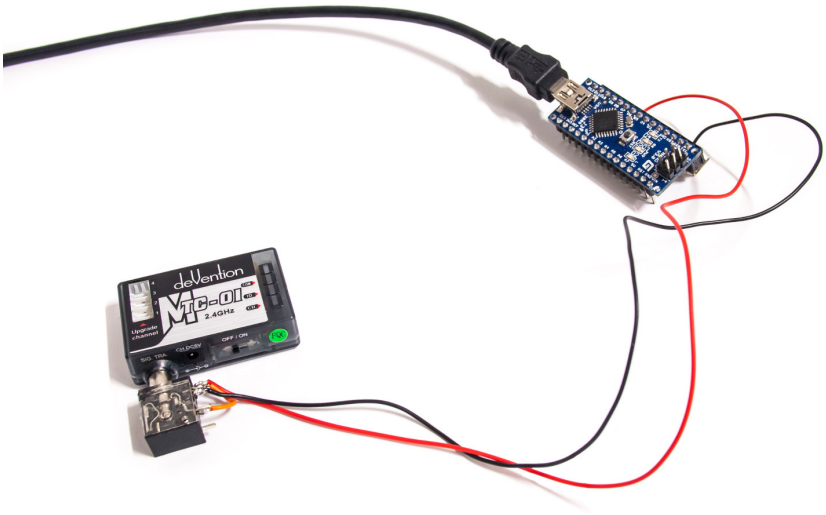


Figure B.2.: Arduino with MagicCube remote

on an IMU. This control loop was implemented in hardware by the manufacturer. Because of the hardware implementations, this control loop probably runs at $50\,\mathrm{Hz}$ to $200\,\mathrm{Hz}$. The pose is controlled using information sensed by the tracking system using simple proportional-integral-derivative (PID) controllers running at $10\,\mathrm{Hz}$. In principle, more complex controllers would be necessary to account for the complex dynamics of the robot but unfortunately, the low-level attitude control loop cannot be turned off. This would be mainly needed for aggressive flight maneuvers. However, for relatively slow movements the hierarchical control formed by the PID controllers and the low-level attitude control is adequate. In a hovering experiment, standard deviations were measured to be of the order of $3\,\mathrm{cm}$ for the lateral position, $1\,\mathrm{cm}$ for height, and $1.5°$ for heading.

B.3.2. Safety Cage

To enhance the safety of bystanders and other scientists not participating in the experiments, a virtual safety cage was implemented. An implementation of the safety cage is depicted in fig. B.3. This virtual safety cage is of size $4\,\mathrm{m} \times 4\,\mathrm{m} \times 2\,\mathrm{m}$, which corresponds to the blue cuboid in the figure. The main function of this cage is to shut down the motors of the robot if it leaves the cage by accident. In contrast to other indoor experiments with flying robots a physical, safety cage or net is not needed because of the small size of the robot.

Figure B.3.: Virtual safety cage

Trajectory Following

Trajectory following was implemented by using waypoints on the desired trajectory as set-points for the PID pose controllers. A simple algorithm iterated through the waypoints if the robot arrived at a position closer than 10 cm to the current waypoint.

As an example, a circular trajectory is shown in fig. B.4 with overlaid snapshots from a video showing the positions of the robot over time. For this circle, 20 waypoints were used.

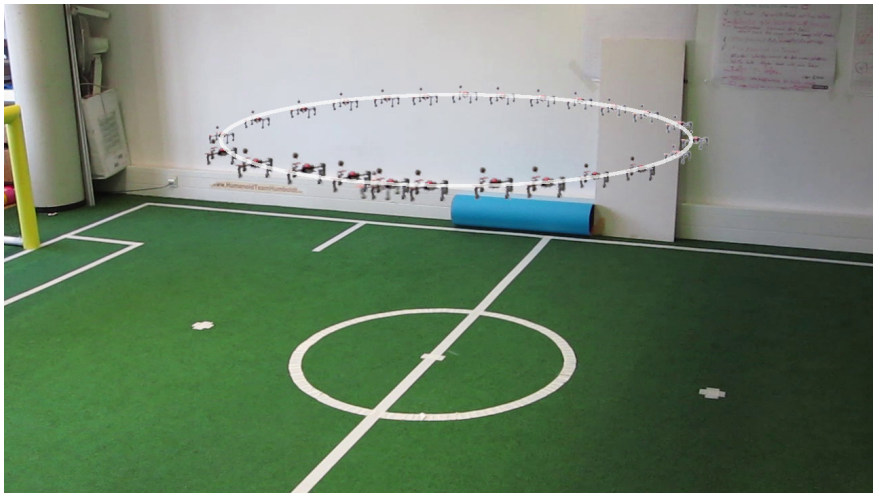


Figure B.4.: The robot follows an exemplary trajectory

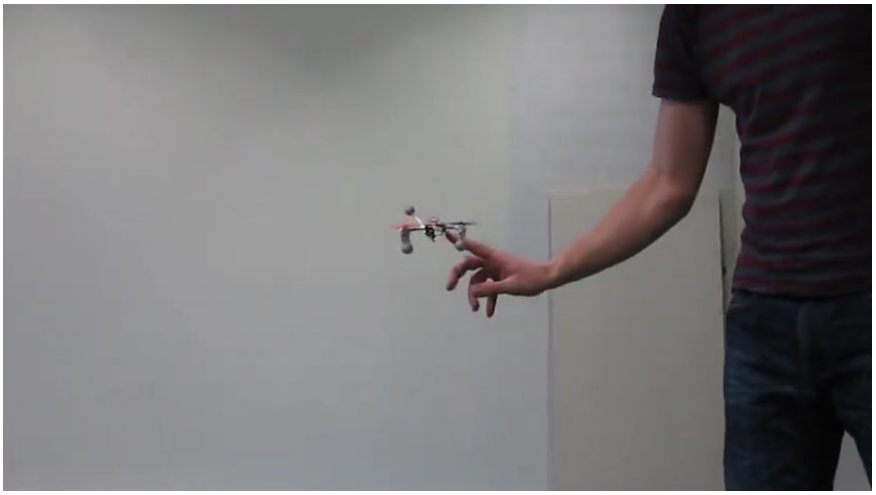


Figure B.5.: Human robot interaction by direct physical interaction, in this case mediated by pushing

B.4. Direct Physical Interaction

A prominent vision is to facilitate interaction between flying robots and humans, that goes beyond device-based remote control but instead emphasizes direct interaction with the robot. This provides a basis for working together with the robot, teaching it by demonstration and physically correcting its mistakes.

Since the setup is working with a tracking system, interaction could be achieved by tracking the human as done in motion capturing systems and working with the modeled data. Other alternatives would be to use a depth-sensing device like Kinect of a stereo camera. The most intuitive way of interaction, however, is direct physical interaction.

This is not only the most intuitive way of interacting with a robot, it also does not require additional hardware, as opposed to also tracking the human or using an RGB-D camera. For this to work, safety is imperative in order to ensure the health of the human and the physical integrity of the robot. An example interaction is depicted in fig. B.5.

The error as determined by the control loop can be used to detect sufficiently large offsets in position, which are due to external forces inflicted on the system by physical contact. A more sensitive alternative is the use of more advanced internal models, and detecting the interaction by monitoring the prediction errors of the forward model. These methods allow us to change the setpoints of the controllers and to freely position the robot in 3D space.

Bibliography

Abaid, N., Marras, S., Fitzgibbons, C., and Porfiri, M. (2013). Modulation of risk-taking behaviour in golden shiners (notemigonus crysoleucas) using robotic fish. *Behavioural processes*, 100:9–12.

Adler, J. (1966). Chemotaxis in bacteria. *Science*, 153:708–716.

Agmon, N. and Peleg, D. (2006). Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM Journal on Computing*, 36(1):56–82.

Andrieu, C., Doucet, A., and Holenstein, R. (2010). Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(3):269–342.

Asimov, I. (1950). *I, ROBOT*. Gnome Press.

Atanasov, N., Le Ny, J., Michael, N., and Pappas, G. J. (2012). Stochastic source seeking in complex environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3013–3018. IEEE.

Ballerini, M., Cabibbo, N., Candelier, R., Cavagna, A., Cisbani, E., Giardina, I., Orlandi, A., Parisi, G., Procaccini, A., Viale, M., et al. (2008). Empirical investigation of starling flocks: a benchmark study in collective animal behaviour. *Animal behaviour*, 76(1):201–215.

Baranes, A. and Oudeyer, P.-Y. (2013). Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73.

Barbosa, M., Bernardino, A., Figueira, D., Gaspar, J., Gonçalves, N., Lima, P. U., Moreno, P., Pahliani, A., Santos-Victor, J., Spaan, M. T., et al. (2009). Isrobotnet: A testbed for sensor and robot network systems. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2827–2833. IEEE.

Barsalou, L. W. (2008). Grounded cognition. *Annu. Rev. Psychol.*, 59:617–645.

Barsalou, L. W. (2009). Simulation, situated conceptualization, and prediction. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1521):1281–1289.

Barsalou, L. W. (2010). Grounded cognition: past, present, and future. *Topics in Cognitive Science*, 2(4):716–724.

Bekmezci, I., Sahingoz, O. K., and Temel, Ş. (2013). Flying ad-hoc networks (fanets): a survey. *Ad Hoc Networks*, 11(3):1254–1270.

Beni, G. (2005). From swarm intelligence to swarm robotics. In *Swarm Robotics*, pages 1–9. Springer.

Bentley, J. L. (1975). Survey of techniques for fixed radius near neighbor searching. Technical report, Stanford Linear Accelerator Center, Calif.(USA).

Berg, H. C. and Brown, D. A. (1972). Chemotaxis in escherichia coli analysed by three-dimensional tracking. *Nature*, 239(5374):500–504.

Bergstra, J., Yamins, D., and Cox, D. D. (2013). Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In van der Walt, S., Millman, J., and Huff, K., editors, *Proceedings of the 12th Python in Science Conference*, pages 13 – 20.

Bergstra, J. S., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554.

Bjerknes, J. D. and Winfield, A. F. (2013). On fault tolerance and scalability of swarm robotic systems. In *Distributed Autonomous Robotic Systems*, pages 431–444. Springer.

Blakemore, S.-J., Wolpert, D., and Frith, C. (2000). Why can't you tickle yourself? *Neuroreport*, 11(11):R11–R16.

Blakemore, S.-J., Wolpert, D. M., and Frith, C. D. (1998). Central cancellation of self-produced tickle sensation. *Nature neuroscience*, 1(7):635–640.

Blobel, V. and Lohrmann, E. (1998). *Statistische und numerische Methoden der Datenanalyse*. Teubner.

Blum, C., Berthold, O., Rhan, P., and Hafner, V. V. (2014). Intuitive control of small flying robots. In *Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction*, HRI '14, pages 128–129, New York, NY, USA. ACM.

Blum, C. and Hafner, V. V. (2012). An autonomous flying robot for network robotics. *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, pages 1 –5.

Blum, C. and Hafner, V. V. (2013). Robust exploration strategies for a robot exploring a wireless network. *Electronic Communications of the EASST*, 56.

Blum, C. and Hafner, V. V. (2014). Gradient-based taxis algorithms for network robotics. *arXiv preprint arXiv:1409.7580*.

Blum, C. and Hafner, V. V. (2015). Active exploration of sensor networks from a robotics perspective. in preparation.

Blum, C., Winfield, A. F. T., and Hafner, V. V. (2015). Internal model based safety. in preparation.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Proceedings volume in the Santa Fe Institute studies in the sciences of complexity. OUP USA.

Bongard, J., Zykov, V., and Lipson, H. (2006). Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121.

Bouzid, Z., Potop-Butucaru, M. G., and Tixeuil, S. (2009). Byzantine-resilient convergence in oblivious robot networks. In *Distributed Computing and Networking*, pages 275–280. Springer.

Bouzid, Z., Potop-Butucaru, M. G., and Tixeuil, S. (2010). Optimal byzantine-resilient convergence in uni-dimensional robot networks. *Theoretical Computer Science*, 411(34):3154–3168.

Braitenberg, V. (1986). *Vehicles: Experiments in synthetic psychology*. MIT press.

Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41.

Briod, A., Kornatowski, P., Klaptocz, A., Garnier, A., Pagnamenta, M., Zufferey, J.-C., and Floreano, D. (2013a). Contact-based navigation for an autonomous flying robot. In *International Conference on Intelligent Robots and Systems (IROS'13)*, pages 3987–3992.

Briod, A., Zufferey, J.-C., and Floreano, D. (2013b). Optic-flow based control of a 46g quadrotor. In *Workshop on Vision-based Closed-Loop Control and Navigation of Micro Helicopters in GPS-denied Environments, IROS'13*.

Bronstein, I. and Semendjajew, K. (2008). *Taschenbuch der Mathematik*. Harri Deutsch.

Brown, T. X., Argrow, B., Dixon, C., Doshi, S., Thekkekunnel, R.-G., and Henkel, D. (2004). Ad hoc uav ground network (augnet). In *AIAA 3rd Unmanned Unlimited Technical Conference*, pages 1–11.

Buhl, J., Sumpter, D. J., Couzin, I. D., Hale, J. J., Despland, E., Miller, E., and Simpson, S. J. (2006). From disorder to order in marching locusts. *Science*, 312(5778):1402–1406.

Cameron, S., Hailes, S., Julier, S., McClean, S., Parr, G., Trigoni, N., Ahmed, M., McPhillips, G., De Nardi, R., Nie, J., Symington, A., Teacy, L., and Waharte, S. (2010). Suaave: Combining aerial robots and wireless networking. In *25th Bristol International UAV Systems Conference*, pages 1–14.

Chaimowicz, L., Cowley, A., Gomez-Ibanez, D., Grocholsky, B., Hsieh, M., Hsu, H., Keller, J., Kumar, V., Swaminathan, R., and Taylor, C. (2005). Deploying air-ground multi-robot teams in urban environments. In *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*, pages 223–234. Springer.

Clark, C. M., Frew, E. W., Jones, H. L., and Rock, S. M. (2003). An integrated system for command and control of cooperative robotic systems. In *International Conference on Advanced Robotics*.

Clement, J., Défago, X., Gradinariu Potop-Butucaru, M., Messika, S., and Raipin-Parvedy, P. (2012). Fault and byzantine tolerant self-stabilizing mobile robots gathering.

Cohen, R. and Peleg, D. (2006). Convergence of autonomous mobile robots with inaccurate sensors and movements. In *STACS 2006*, pages 549–560. Springer.

Conant, R. C. and Ross Ashby, W. (1970). Every good regulator of a system must be a model of that system? *International journal of systems science*, 1(2):89–97.

Correll, N., Bachrach, J., Vickery, D., and Rus, D. (2009). Ad-hoc wireless network coverage with networked robots that cannot localize. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3878–3885. IEEE.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Courant, R., Friedrichs, K., and Lewy, H. (1928). Über die partiellen Differenzengleichungen der mathematischen Physik. *Mathematische Annalen*, 100(1):32–74.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27.

Craik, K. J. W. (1967). *The Nature of Explanation*. Cambridge University Press.

Cremean, L., Dunbar, W. B., van Gogh, D., Hickey, J., Klavins, E., Meltzer, J., and Murray, R. M. (2002). The caltech multi-vehicle wireless testbed. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 1, pages 86–88. IEEE.

Cruz, D., McClintock, J., Perteet, B., Orqueda, O. A., Cao, Y., and Fierro, R. (2007). Decentralized cooperative control-a multivehicle platform for research in networked embedded systems. *Control Systems, IEEE*, 27(3):58–78.

Damosso, E. (1998). Digital mobile radio: Cost 231 view on the evolution towards 3rd generation systems. *Final Report of the COST 231 Project*.

Damosso, E. and Correia, L. M. (1999). *COST Action 231: Digital Mobile Radio Towards Future Generation Systems: Final Report*. European Commission.

Daniel, K., Rohde, S., Goddemeier, N., and Wietfeld, C. (2010). A communication aware steering strategy avoiding self-separation of flying robot swarms. In *Intelligent Systems (IS), 2010 5th IEEE International Conference*, pages 254–259. IEEE.

Dantu, K., Goyal, P., and Sukhatme, G. (2009). Relative bearing estimation from commodity radios. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3871–3877. IEEE.

Dantu, K., Rahimi, M., Shah, H., Babel, S., Dhariwal, A., and Sukhatme, G. S. (2005). Robomote: enabling mobility in sensor networks. In *Proceedings of the 4th international symposium on Information processing in sensor networks*, page 55. IEEE Press.

Dasgupta, D., Yu, S., and Nino, F. (2011). Recent advances in artificial immune systems: models and applications. *Applied Soft Computing*, 11(2):1574–1587.

Davidson, D. B. (2005). *Computational electromagnetics for RF and microwave engineering*. Cambridge University Press.

De, P., Raniwala, A., Krishnan, R., Tatavarthi, K., Modi, J., Syed, N. A., Sharma, S., and Chiueh, T.-c. (2006). Mint-m: an autonomous mobile wireless experimentation platform. In *Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 124–137. ACM.

Dearden, A. (2008). *Developmental learning of internal models for robotics*. PhD thesis, Imperial College London.

Dearden, A. and Demiris, Y. (2005). Learning forward models for robots. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 1440–1445, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Demiris, Y. (2007). Prediction of intent in robotics and multi-agent systems. *Cognitive processing*, 8(3):151–158.

Demiris, Y. and Dearden, A. (2005). From motor babbling to hierarchical learning by imitation: a robot developmental pathway. In *Proceedings of the 5th International Workshop on Epigenetic Robotics*, pages 31–37. Lund University Cognitive Studies.

Demiris, Y. and Khadhouri, B. (2006). Hierarchical attentive multiple models for execution and recognition of actions. *Robotics and autonomous systems*, 54(5):361–369.

Der, R. and Martius, G. (2006). From motor babbling to purposive actions: Emerging self-exploration in a dynamical systems approach to early robot development. In *From Animals to Animats 9*, pages 406–421. Springer.

Derenick, J., Fink, J., and Kumar, V. (2011). Localization using ambiguous bearings from radio signal strength. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3248–3253. IEEE.

Dixon, C. (2010). *Controlled mobility of unmanned aircraft chains to optimize network capacity in realistic communication environments*. PhD thesis, University of Colorado at Boulder.

Echeverria, G., Lassabe, N., Degroote, A., and Lemaignan, S. (2011). Modular open robots simulation engine: Morse. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 46–51. IEEE.

Edlund, E., Lindgren, O., and Jacobi, M. N. (2011). Designing isotropic interactions for self-assembly of complex lattices. *Physical review letters*, 107(8):085503.

Elnahrawy, E., Austen-Francisco, J., and Martin, R. P. (2007). Adding angle of arrival modality to basic rss location management techniques. In *Wireless Pervasive Computing, 2007. ISWPC'07. 2nd International Symposium on*. IEEE.

Emelyanov, S. (1970). Automatic control systems with variable structure. Technical report, DTIC Document.

Faria, J. J., Dyer, J. R., Clément, R. O., Couzin, I. D., Holt, N., Ward, A. J., Waters, D., and Krause, J. (2010). A novel method for investigating the collective behaviour of fish: introducing robofish. *Behavioral Ecology and Sociobiology*, 64(8):1211–1218.

Fink, J. and Kumar, V. (2010). Online methods for radio signal mapping with mobile robots. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1940–1945. IEEE.

Fish, R., Flickinger, M., and Lepreau, J. (2006). Mobile emulab: A robotic wireless and sensor network testbed. In *IEEE INFOCOM*.

Flanagan, J. R., King, S., Wolpert, D. M., and Johansson, R. S. (2001). Sensorimotor prediction and memory in object manipulation. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, 55(2):87.

Francis, B. A. and Wonham, W. M. (1976). The internal model principle of control theory. *Automatica*, 12(5):457–465.

Garcia, C. E. and Morari, M. (1982). Internal model control. a unifying review and some new results. *Industrial & Engineering Chemistry Process Design and Development*, 21(2):308–323.

Gauss, C. F. (1823). *Theoria combinationis observationum erroribus minimis obnoxiae*. H. Dieterich.

Goldsmith, A. (2005). *Wireless Communications*. Cambridge University Press.

Gosmann, J., Blum, C., Berthold, O., and Hafner, V. V. (2013). Tactile sensors for learning of soft landing on a flying robot. In *Workshop: Sensitive Robotics, Robotics: Science and Systems 2013*.

Grèzes, J., Armony, J. L., Rowe, J., and Passingham, R. E. (2003). Activations related to "mirror" and "canonical" neurones in the human brain: an fmri study. *Neuroimage*, 18(4):928–937.

Group, I. . W. et al. (2010). Ieee standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements–part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 6: Wireless access in vehicular environments. *IEEE Std*, 802:11p.

Gupta, P. and Kumar, P. R. (2000). The capacity of wireless networks. *Information Theory, IEEE Transactions on*, 46(2):388–404.

Halloy, J., Sempo, G., Caprari, G., Rivault, C., Asadpour, M., Tâche, F., Said, I., Durier, V., Canonge, S., Amé, J. M., et al. (2007). Social integration of robots into groups of cockroaches to control self-organized choices. *Science*, 318(5853):1155–1158.

Han, D., Andersen, D. G., Kaminsky, M., Papagiannaki, K., and Seshan, S. (2009). Access point localization using local signal strength gradient. *Passive and Active Network Measurement*, pages 99–108.

Haruno, M., Wolpert, D., and Kawato, M. (2001). Mosaic model for sensorimotor learning and control. *Neural computation*, 13(10):2201–2220.

Haruno, M., Wolpert, D. M., and Kawato, M. (1999). Multiple paired forward-inverse models for human motor learning and control. *Advances in neural information processing systems*, pages 31–37.

Hauert, S., Leven, S., Zufferey, J.-C., and Floreano, D. (2010a). Communication-based leashing of real flying robots. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 15–20. IEEE.

Hauert, S., Leven, S., Zufferey, J.-C., and Floreano, D. (2010b). Communication-based swarming for flying robots. In *Proceedings of the Workshop on Network Science and Systems Issues in Multi-Robot Autonomy, IEEE International Conference on Robotics and Automation*. Ieee Service Center, 445 Hoes Lane, Po Box 1331, Piscataway, Nj 08855-1331 Usa.

Hauert, S., Leven, S., Zufferey, J.-C., and Floreano, D. (2010c). Communication-based swarming for flying robots. In *International Workshop on Self-Organized Systems*, number LIS-POSTER-2010-001.

Hauert, S., Zufferey, J.-C., and Floreano, D. (2009). Evolved swarming without positioning information: an application in aerial communication relay. *Autonomous Robots*, 26(1):21–32.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*.

Helbing, D., Farkas, I., and Vicsek, T. (2000). Simulating dynamical features of escape panic. *Nature*, 407(6803):487–490.

Hemelrijk, C. K. and Hildenbrandt, H. (2008). Self-organized shape and frontal density of fish schools. *Ethology*, 114(3):245–254.

Hemelrijk, C. K. and Kunz, H. (2005). Density distribution and size sorting in fish schools: an individual-based model. *Behavioral Ecology*, 16(1):178–187.

Heo, N. and Varshney, P. K. (2005). Energy-efficient deployment of intelligent mobile sensor networks. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 35(1):78–92.

Hesslow, G. (2012). The current status of the simulation theory of cognition. *Brain research*, 1428:71–79.

Higgins, F., Tomlinson, A., and Martin, K. M. (2009). Survey on security challenges for swarm robotics. In *Autonomic and Autonomous Systems, 2009. ICAS'09. Fifth International Conference on*, pages 307–312. IEEE.

Hildenbrandt, H., Carere, C., and Hemelrijk, C. K. (2010). Self-organized aerial displays of thousands of starlings: a model. *Behavioral Ecology*, 21(6):1349–1359.

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Hoffmann, H. (2007). Perception through visuomotor anticipation in a mobile robot. *Neural Networks*, 20(1):22–33.

Hoffmann, H. and Möller, R. (2004). Action selection and mental transformation based on a chain of forward models. *From Animals to Animats*, 8:213–222.

Holland, O. and Goodman, R. (2003). Robots with internal models a route to machine consciousness? *Journal of Consciousness Studies*, 10(4-5):77–109.

Howard, A., Matarić, M. J., and Sukhatme, G. S. (2002). Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Distributed Autonomous Robotic Systems 5*, pages 299–308. Springer.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in science and engineering*, 9(3):90–95.

Ito, M. (2008). Control of mental activities by internal models in the cerebellum. *Nature Reviews Neuroscience*, 9(4):304–313.

Jackson, J. D. (1962). *Classical electrodynamics*. Wiley.

Jacob, P. and Jeannerod, M. (2005). The motor theory of social cognition: a critique. *Trends in cognitive sciences*, 9(1):21–25.

Jacobi, N., Husbands, P., and Harvey, I. (1995). Noise and the reality gap: The use of simulation in evolutionary robotics. In *Proceedings of the Third European Conference on Advances in Artificial Life,*, pages 704–720. Springer.

Jiménez-González, A., Martínez-de Dios, J. R., and Ollero, A. (2011). An integrated testbed for cooperative perception with heterogeneous mobile and static sensors. *Sensors*, 11(12):11516–11543.

Jiménez-González, A., Martinez-de Dios, J. R., and Ollero, A. (2013). Testbeds for ubiquitous robotics: A survey. *Robotics and Autonomous Systems*, 61(12):1487–1501.

Jin, Z., Waydo, S., Wildanger, E. B., Lammers, M., Scholze, H., Foley, P., Held, D., and Murray, R. M. (2004). Mvwt-ii: The second generation caltech multi-vehicle wireless testbed. In *American Control Conference, 2004. Proceedings of the 2004*, volume 6, pages 5321–5326. IEEE.

Jones, E., Oliphant, T., Peterson, P., et al. (2001–). SciPy: Open source scientific tools for Python. [Online; accessed 2015-05-19].

Jones, J. A. and Keough, D. (2008). Auditory-motor mapping for pitch control in singers and nonsingers. *Experimental brain research*, 190(3):279–287.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45.

Kawato, M. (1999). Internal models for motor control and trajectory planning. *Current opinion in neurobiology*, 9(6):718–727.

Keysers, C. and Gazzola, V. (2010). Social neuroscience: mirror neurons recorded in humans. *Current Biology*, 20(8):R353–R354.

Kilner, J. M. (2011). More than one pathway to action understanding. *Trends in cognitive sciences*, 15(8):352–357.

Klaptocz, A., Briod, A., Daler, L., Zufferey, J.-C., and Floreano, D. (2013). Euler spring collision protection for flying robots. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1886–1892. IEEE.

Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE.

Kohler, E., Morris, R., Chen, B., Jannotti, J., and Kaashoek, M. F. (2000). The click modular router. *ACM Transactions on Computer Systems (TOCS)*, 18(3):263–297.

Krishnamachari, B., Wicker, S., Bejar, R., and Fernandez, C. (2003). On the complexity of distributed self-configuration in wireless networks. *Telecommunication Systems*, 22(1-4):33–59.

Kushner, H. and Yin, G. (2003). *Stochastic Approximation and Recursive Algorithms and Applications*. Springer.

Lakoff, G. (2014). Mapping the brain's metaphor circuitry: metaphorical thought in everyday reason. *Frontiers in human neuroscience*, 8.

Lau, H., Bate, I., Cairns, P., and Timmis, J. (2011a). Adaptive data-driven error detection in swarm robotics with statistical classifiers. *Robotics and Autonomous Systems*, 59(12):1021–1035.

Lau, H., Bate, I., and Timmis, J. (2013). Immune-inspired error detection for multiple faulty robots in swarm robotics. In *Advances in Artificial Life, ECAL*, volume 12, pages 846–853.

Lau, H., Timmis, J., and Bate, I. (2011b). Collective self-detection scheme for adaptive error detection in a foraging swarm of robots. In *Artificial Immune Systems*, pages 254–267. Springer.

LeCun, Y. (1985). Une procédure d'apprentissage pour réseau à seuil asymétrique (a learning scheme for asymmetric threshold networks). In *Proceedings of Cognitiva*, volume 85, pages 599–604.

Lee, D. et al. (2013). Semiautonomous haptic teleoperation control architecture of multiple unmanned aerial vehicles. *Mechatronics, IEEE/ASME Transactions on*, 18(4):1334–1345.

Little, D. Y. and Sommer, F. T. (2013). Learning and exploration in action-perception loops. *Frontiers in Neural Circuits*, 7(37).

Liu, J. S. and Chen, R. (1998). Sequential monte carlo methods for dynamic systems. *Journal of the American statistical association*, 93(443):1032–1044.

Liu, W. and Winfield, A. F. T. (2011). Open-hardware e-puck Linux extension board for experimental swarm robotics research. *Microprocessors and Microsystems*, 35(1).

Lux, R. and Shi, W. (2004). Chemotaxis-guided movements in bacteria. *Critical Reviews in Oral Biology & Medicine*, 15(4):207–220.

Mahasukhon, P., Hempel, M., Sharif, H., Zhou, T., Ci, S., and Chen, H.-H. (2007). Ber analysis of 802.11 b networks under mobility. In *Communications, 2007. ICC'07. IEEE International Conference on*, pages 4722–4727. IEEE.

Mao, G., Fidan, B., and Anderson, B. D. (2007). Wireless sensor network localization techniques. *Computer networks*, 51(10):2529–2553.

Mardia, K. V., Kent, J. T., and Bibby, J. M. (1979). *Multivariate analysis*. Academic press.

Marques, H. G. and Holland, O. (2009). Architectures for functional imagination. *Neurocomputing*, 72(4):743–759.

Martius, G., Jahn, L., Hauser, H., and Hafner, V. V. (2014). Self-exploration of the stumpy robot with predictive information maximization. In del Pobil, A. P., Chinellato, E., Martinez-Martin, E., Hallam, J., Cervera, E., and Morales, A., editors, *From Animals to Animats 13*, volume 8575 of *Lecture Notes in Computer Science*, pages 32–42. Springer International Publishing.

Mattern, F. and Floerkemeier, C. (2010). From the internet of computers to the internet of things. In *From active data management to event-based systems and more*, pages 242–259. Springer.

McKinney, W. (2011). pandas: a foundational python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, pages 1–9.

McLain, T. W. and Beard, R. W. (2004). Unmanned air vehicle testbed for cooperative control experiments. In *Proceedings of the American Control Conference*, pages 5327–5331.

Merfeld, D. M., Zupan, L., and Peterka, R. J. (1999). Humans use internal models to estimate gravity and linear acceleration. *Nature*, 398(6728):615–618.

Michel, O. (2004). Webots: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):39–42.

Milic, B. and Malek, M. (2009). Npart - node placement algorithm for realistic topologies in wireless multihop network simulation. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, Simutools '09, pages 9:1–9:10, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Millard, A. G., Timmis, J., and Winfield, A. F. (2014a). Run-time detection of faults in autonomous mobile robots based on the comparison of simulated and real robot behaviour. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 3720–3725. IEEE.

Millard, A. G., Timmis, J., and Winfield, A. F. (2014b). Towards exogenous fault detection in swarm robotic systems. In *Towards Autonomous Robotic Systems*, pages 429–430. Springer.

Mills, K. L. (2007). A brief survey of self-organization in wireless sensor networks. *Wireless Communications and Mobile Computing*, 7(7):823–834.

Mischiati, M., Lin, H.-T., Herold, P., Imler, E., Olberg, R., and Leonardo, A. (2015). Internal models direct dragonfly interception steering. *Nature*, 517(7534):333–338.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Möller, R. (2009). Local visual homing by warping of two-dimensional images. *Robotics and Autonomous Systems*, 57(1):87–101.

Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D., and Martinoli, A. (2009). The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions*, volume 1, pages 59–65. IPCB: Instituto Politécnico de Castelo Branco.

Morari, M. and Lee, J. H. (1999). Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4):667–682.

Moraud, E. and Martinez, D. (2010). Effectiveness and robustness of robot infotaxis for searching in dilute conditions. *Frontiers in Neurorobotics*, 4(1).

Nembrini, J., Winfield, A., and Melhuish, C. (2002). Minimalist coherent swarming of wireless networked autonomous mobile robots. In *From Animals to Animats 7: Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior*, volume 7, page 373. MIT Press.

Nguyen-Tuong, D. and Peters, J. (2011). Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–340.

Obraczka, K., Boice, J., Martínez-Gómez, L., Francois, J. P., Levin-Pick, A., and Weitzenfeld, A. (2007). Star: Ad-hoc wireless networking for autonomous multi-robot coordination. *Proceedings of the Robocomm*.

O'Dowd, P. J., Winfield, A. F. T., and Studley, M. (2011). The distributed co-evolution of an embodied simulator and controller for swarm robot behaviours. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 4995–5000.

O'Keefe, J. (1976). Place units in the hippocampus of the freely moving rat. *Experimental neurology*, 51(1):78–109.

Oskooi, A. F., Roundy, D., Ibanescu, M., Bermel, P., Joannopoulos, J. D., and Johnson, S. G. (2010). MEEP: A flexible free-software package for electromagnetic simulations by the FDTD method. *Computer Physics Communications*, 181:687–702.

Oztop, E., Kawato, M., and Arbib, M. (2006). Mirror neurons and imitation: A computationally guided review. *Neural Networks*, 19(3):254–271.

Patwari, N., Ash, J., Kyperountas, S., Hero, A.O., I., Moses, R., and Correal, N. (2005). Locating the nodes: cooperative localization in wireless sensor networks. *Signal Processing Magazine, IEEE*, 22(4):54 – 69.

Paul, L. Y., Twigg, J. N., and Sadler, B. M. (2011). Radio signal strength tracking and control for robotic networks. In *SPIE Defense, Security, and Sensing*, pages 803116–803116. International Society for Optics and Photonics.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pettersson, O. (2005). Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems*, 53(2):73–88.

Pezzulo, G. (2011). Grounding procedural and declarative knowledge in sensorimotor anticipation. *Mind & Language*, 26(1):78–114.

Pezzulo, G., Barsalou, L. W., Cangelosi, A., Fischer, M. H., McRae, K., and Spivey, M. J. (2011). The mechanics of embodiment: A dialog on embodiment and computational modeling. *Embodied and grounded cognition*, page 196.

Pfeifer, R. and Bongard, J. (2006). *How the body shapes the way we think: a new view of intelligence*. MIT press.

Pfeifer, R., Lungarella, M., and Iida, F. (2007). Self-organization, embodiment, and biologically inspired robotics. *science*, 318(5853):1088–1093.

Pitman, D. and Cummings, M. L. (2012). Collaborative exploration with a micro aerial vehicle: A novel interaction method for controlling a mav with a hand-held device. *Advances in Human-Computer Interaction*.

Premack, D. and Woodruff, G. (1978). Does the chimpanzee have a theory of mind? *Behavioral and brain sciences*, 1(04):515–526.

Prencipe, G. (2001). Corda: Distributed coordination of a set of autonomous mobile robots. In *Proceedings of the 4th European Research Seminar on Advances in Distributed Systems (ERSADS 2001)*, pages 185–190.

Qin, L., He, X., and Zhou, D. (2014). A survey of fault diagnosis for swarm systems. *Systems Science & Control Engineering: An Open Access Journal*, 2(1):13–23.

Rappaport, T. (2001). *Wireless Communications: Principles and Practice*. Prentice Hall PTR.

Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. Adaptative computation and machine learning series. University Press Group Limited.

Reich, J., Misra, V., and Rubenstein, D. (2008). Roomba madnet: a mobile ad-hoc delay tolerant network testbed. *ACM SIGMOBILE Mobile Computing and Communications Review*, 12(1):68–70.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *ACM SIGGRAPH Computer Graphics*, volume 21, pages 25–34. ACM.

Riedmiller, M. (1994). Advanced supervised learning in multi-layer perceptrons — from backpropagation to adaptive learning algorithms. *Computer Standards & Interfaces*, 16(3):265–278.

Rizzolatti, G. and Craighero, L. (2004). The mirror-neuron system. *Annu. Rev. Neurosci.*, 27:169–192.

Rizzolatti, G., Fadiga, L., Gallese, V., and Fogassi, L. (1996). Premotor cortex and the recognition of motor actions. *Cognitive brain research*, 3(2):131–141.

Rosati, S., Rimoldi, B., Kruzelecki, K., Jimenez-Pacheco, A., Floreano, D., Zufferey, J.-C., and Heitz, G. H. M. (2013). Testbed for fast-deployable flying wifi networks. In *The Fifth Nordic Workshop on System and Network Optimization for Wireless*.

Rosenblueth, A. and Wiener, N. (1945). The role of models in science. *Philosophy of science*, 12(4):316–321.

Rubenstein, M., Ahler, C., and Nagpal, R. (2012). Kilobot: A low cost scalable robot system for collective behaviors. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3293–3298. IEEE.

Rubenstein, M., Cornejo, A., and Nagpal, R. (2014). Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, DTIC Document.

Şahin, E. (2005). Swarm robotics: From sources of inspiration to domains of application. In *Swarm robotics*, pages 10–20. Springer.

Schaul, T., Bayer, J., Wierstra, D., Sun, Y., Felder, M., Sehnke, F., Rückstieß, T., and Schmidhuber, J. (2010). Pybrain. *Journal of Machine Learning Research*, 11:743–746.

Schillaci, G. and Hafner, V. V. (2011). Random movement strategies in self-exploration for a humanoid robot. In *Proceedings of the 6th international conference on Human-robot interaction*, pages 245–246. ACM.

Schillaci, G., Hafner, V. V., and Lara, B. (2012a). Coupled inverse-forward models for action execution leading to tool-use in a humanoid robot. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 231–232. ACM.

Schillaci, G., Hafner, V. V., Lara, B., and Grosjean, M. (2013). Is that me?: sensorimotor learning and self-other distinction in robotics. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pages 223–224. IEEE Press.

Schillaci, G., Lara, B., and Hafner, V. V. (2012b). Internal simulations for behaviour selection and recognition. In *Human Behavior Understanding*, pages 148–160. Springer.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.

Schölkopf, B. and Smola, A. J. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press.

Shannon, C. E. (1951). Presentation of a maze-solving machine. In *Proceedings of the Eighth Conf. of the Josiah Macy Jr. Found.*, Cybernetics, pages 173–108.

Smith, S. (1997). *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Pub.

Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222.

Sommer, M. A. and Wurtz, R. H. (2008). Brain circuits for the internal monitoring of movements. *Annual review of neuroscience*, 31:317.

Souissi, S., Yang, Y., and Défago, X. (2008). Fault-tolerant flocking in a k-bounded asynchronous system. In *Principles of Distributed Systems*, pages 145–163. Springer.

Spall, J. (2005). *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley.

Spears, W. M. (2012). *Physicomimetics: Physics-based swarm intelligence*. Springer.

STMicroelectronics (2011). AN3359 Application note: Low cost PCB antenna for 2.4GHz radio: Meander design.

Stoodley, C. J. (2012). The cerebellum and cognition: evidence from functional imaging studies. *The Cerebellum*, 11(2):352–365.

Sutton, R. S. and Barto, A. G. (1998). *Introduction to reinforcement learning*. MIT Press.

Taflove, A. and Hagness, S. C. (2005). *Computational Electrodynamics*. Artech house.

Tanenbaum, A. and Wetherall, D. (2012). *Computernetzwerke*. Pearson.

Taube, J. S., Muller, R. U., and Ranck, J. B. (1990). Head-direction cells recorded from the postsubiculum in freely moving rats. i. description and quantitative analysis. *The Journal of Neuroscience*, 10(2):420–435.

Tikhonov, A. N. (1943). On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*, volume 39, pages 195–198.

Timmis, J., Andrews, P., and Hart, E. (2010). On artificial immune systems and swarm intelligence. *Swarm Intelligence*, 4(4):247–273.

Tin, C. and Poon, C.-S. (2005). Internal models in sensorimotor integration: perspectives from adaptive control theory. *Journal of neural engineering*, 2(3):S147.

Tulving, E. (1985). *Elements of episodic memory*. Oxford University Press.

Twigg, J. N., Fink, J. R., Yu, P., and Sadler, B. M. (2012). Rss gradient-assisted frontier exploration and radio source localization. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 889–895. IEEE.

Vapnik, V. (1963). Pattern recognition using generalized portrait method. *Automation and remote control*, 24:774–780.

Vapnik, V. N. and Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280.

Vaughan, R. (2008). Massively multi-robot simulation in stage. *Swarm Intelligence*, 2(2-4):189–208.

Vaughan, R. T. and Gerkey, B. P. (2007). Really reused robot code from the player/stage project. In Brugali, D., editor, *Software Engineering for Experimental Robotics*, pages 267–289. Springer.

Vaughan, R. T. and Zuluaga, M. (2006). Use your illusion: Sensorimotor self-simulation allows complex agents to plan with incomplete self-knowledge. In *Proc. International Conference on Simulation of Adaptive Behaviour (SAB)*, pages 298–309.

Vicsek, T., Czirók, A., Ben-Jacob, E., Cohen, I., and Shochet, O. (1995). Novel type of phase transition in a system of self-driven particles. *Physical review letters*, 75(6):1226.

Wadhwa, A., Madhow, U., Hespanha, J., and Sadler, B. M. (2011). Following an rf trail to its source. In *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on*, pages 580–587. IEEE.

Wallach, W. and Allen, C. (2009). *Moral Machines: Teaching Robots Right from Wrong*. Oxford: Oxford University Press.

Wang, B., Lim, H. B., and Ma, D. (2009). A survey of movement strategies for improving network coverage in wireless sensor networks. *Computer Communications*, 32(13):1427–1436.

Wang, G., Cao, G., and La Porta, T. (2006). Movement-assisted sensor deployment. *Mobile Computing, IEEE Transactions on*, 5(6):640–652.

Wang, J., Ghosh, R. K., and Das, S. K. (2010). A survey on sensor localization. *Journal of Control Theory and Applications*, 8(1):2–11.

Webb, B. (2004). Neural mechanisms for prediction: do insects have forward models? *Trends in neurosciences*, 27(5):278–282.

Weiss, C., Herwig, A., and Schütz-Bosbach, S. (2011). The self in action effects: Selective attenuation of self-generated sounds. *Cognition*, 121(2):207–218.

Werbos, P. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University.

Winfield, A. F. and Nembrini, J. (2006). Safety in numbers: fault-tolerance in robot swarms. *International Journal of Modelling, Identification and Control*, 1(1):30–37.

Winfield, A. F. T. (2014). *Robots with Internal Models: A Route to Self-Aware and Hence Safer Robots*, chapter 16, pages 237–252. World Scientific.

Winfield, A. F. T., Blum, C., and Liu, W. (2014). Towards an ethical robot: Internal models, consequences and ethical action selection. In *Advances in Autonomous Robotics Systems*, pages 85–96. Springer.

Wolpert, D. M., Diedrichsen, J., and Flanagan, J. R. (2011). Principles of sensorimotor learning. *Nature Reviews Neuroscience*, 12(12):739–751.

Wolpert, D. M., Doya, K., and Kawato, M. (2003). A unifying computational framework for motor control and social interaction. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 358(1431):593–602.

Wolpert, D. M., Ghahramani, Z., and Jordan, M. I. (1995). An internal model for sensorimotor integration. *Science-AAAS-Weekly Paper Edition*, 269(5232):1880–1882.

Wolpert, D. M. and Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7):1317–1329.

Wu, F.-J., Kao, Y.-F., and Tseng, Y.-C. (2011). From wireless sensor networks towards cyber physical systems. *Pervasive and Mobile Computing*, 7(4):397–413.

Yang, Y., Souissi, S., Defago, X., and Takizawa, M. (2009). Fault-tolerant flocking of mobile robots with whole formation rotation. In *Advanced Information Networking and Applications, 2009. AINA'09. International Conference on*, pages 830–837. IEEE.

Yee, K. (1966). Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *IEEE Trans. Antennas Propag*, 14(3):302–307.

Younis, M. and Akkaya, K. (2008). Strategies and techniques for node placement in wireless sensor networks: A survey. *Ad Hoc Networks*, 6(4):621–655.

Younis, M., Senturk, I. F., Akkaya, K., Lee, S., and Senel, F. (2014). Topology management techniques for tolerating node failures in wireless sensor networks: A survey. *Computer Networks*, 58:254–283.

Zago, M., Bosco, G., Maffei, V., Iosa, M., Ivanenko, Y. P., and Lacquaniti, F. (2004). Internal models of target motion: expected dynamics overrides measured kinematics in timing manual interceptions. *Journal of neurophysiology*, 91(4):1620–1634.

Zhu, C., Zheng, C., Shu, L., and Han, G. (2012). A survey on coverage and connectivity issues in wireless sensor networks. *Journal of Network and Computer Applications*, 35(2):619–632.

Ziemke, T., Jirenhed, D.-A., and Hesslow, G. (2005). Internal simulation of perception: a minimal neuro-robotic model. *Neurocomputing*, 68:85–104.

Zubow, A. and Sombrutzki, R. (2011). Evaluation of a lowcost ieee802. 11n mimo testbed. Technical report.

:wq